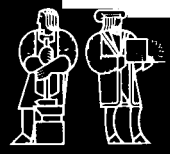
LABORATORY FOR COMPUTER SCIENCE



MASSACHUSETTS INSTITUTE OF TECHNOLOGY

MIT/LCS/TR-465

ANALYSIS OF RANDOM DROP FOR GATEWAY CONGESTION CONTROL

Eman Salaheddin Hashem

November 1989

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

Analysis of Random Drop for Gateway Congestion Control

by

Eman Salaheddin Hashem

Submitted to the
Department of Electrical Engineering and Computer Science
on August 30, 1989 in Partial Fulfillment of the Requirements
for the Degree of Master of Science

This research was supported by the Advanced Research Projects Agency of the Department of Defense, monitored by the National Aeronautics and Space Administration under Contract No. NAG2-582.

Massachusetts Institute of Technology
Laboratory for Computer Science
Cambridge, Massachusetts 02139

Analysis of Random Drop for Gateway Congestion Control

by

Eman Salaheddin Hashem

Submitted to the Department of Electrical Engineering and Computer Science
on August 30, 1989, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

Lately, the growing demand on the Internet has prompted the need for more effective congestion control policies. Currently No Gateway Policy is used to relieve and signal congestion, which leads to unfair service to the individual users and a degradation of overall network performance. ~~This thesis uses~~ network simulation to illustrate the character of Internet congestion and its causes. ~~It considers~~ a newly proposed gateway congestion control policy, called Random Drop, as a promising solution to the pressing problem.

Random Drop relieves resource congestion upon buffer overflow by choosing a random packet from the service queue to be dropped. The random choice should result in a drop distribution proportional to the bandwidth distribution among all contending TCP connections, thus applying the necessary fairness. Nonetheless, the simulation experiments demonstrate several shortcomings with this policy. Because Random Drop is a congestion control policy, which is not applied until congestion has already occurred, it usually results in a high drop rate that hurts too many connections including well-behaved ones. Even though the number of packets dropped is different from one connection to another depending on the buffer utilization upon overflow, the TCP recovery overhead is high enough to neutralize these differences, causing unfair congestion penalities. Besides, the drop distribution itself is an inaccurate representation of the average bandwidth distribution, missing much important information about the bandwidth utilization between buffer overflow events.

A modification of Random Drop to do congestion avoidance by applying the policy early was also proposed. Early Random Drop has the advantage of avoiding the high drop rate of buffer overflow. The early application of the policy removes the pressure of congestion relief and allows more accurate signaling of congestion. To be used effectively, algorithms for the dynamic adjustment of the parameters of Early Random Drop to suite the current network load must still be developed.

Thesis Supervisor: David D. Clark
Title: Senior Research Scientist

Acknowledgements

Many thanks to my thesis supervisor, Dr. David Clark, for all his guidance and assistance throughout my years at MIT. I owe him all of my practical knowledge about the field of networking and computer communication. He was always so generous both with technical and nontechnical advice whenever needed.

Special thanks go to Lixia Zhang and David Feldmeier for their friendship and support. They made the transition into MIT much easier, always supplying technical and nontechnical guidance and answering my questions with great patience. I would also like to thank all members of the Advanced Network Architecture group at MIT Laboratory for Computer Science for the stimulating research atmosphere and the interesting distractions.

And across the oceans, I wish to thank my family for their love and support, especially my mom for her unceasing belief in me. Last but not least, I would never have made it to MIT without the continuous encouragement and love of my dearest and best friend Rami.

To the memory of my father Salaheddin Hashem,
may his soul rest in peace.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the widespread use of computers nowadays comes an increasing demand on computer networks to provide fast cheap communication to the vastly growing research and business communities. As with many other aspects of technology, the high demand is driving the network load to its capacity limit, making congestion control a serious issue. Intelligent congestion control protocols are needed to achieve good performance in dynamic network environments with varying traffic characteristics. This need has recently stirred the Internet research community to review current congestion control schemes and make suggestions for new ones that might be more appropriate for the growing Internet demand.

The Internet is composed of a large number of interconnected networks of various types, usually called TCP/IP networks. These networks, in turn, consist of interconnected communication and processing resources with fixed capacities that support user communication through end-to-end connections. This transport level[1] communication is managed by the Transmission Control Protocol (TCP) [RFC793] which functions on top of a datagram network service managed by the Internet Protocol (IP) [RFC791], thus the name TCP/IP networks.

TCP provides a reliable efficient communication service that is mostly transparent to the user. The TCP and the network resources collaborate to regulate the traffic flow in the network to achieve optimal performance. TCP flow control mechanisms and network congestion control

[1] According to the ISO Open Systems Interconnection Model, the transport level is the fourth from the bottom of the layering hierarchy, lying immediately on top of the basic network level or service.

schemes are continuously being analyzed and evaluated in attempts to improve the quality of service provided to the user. The latest efforts by the Internet research community suggest a preference for a new gateway congestion control policy called Random Drop. This thesis is an analysis of Random Drop and its inherent strengths with the hope that the results will provide enlighting information to any vendors pondering its implementation in their gateways and helpful guidelines to future congestion control policies.

1.1 Flow Control

The Transmission Control Protocol was designed to function on top of a heterogeneous internetwork, with an unreliable datagram network service. TCP provides users with efficient reliable communication, while managing the data flow into the network. It employs window-based flow control to limit the amount of outstanding unacknowledged data under all traffic conditions, but it does not specify a flow control strategy to handle network congestion.

Different TCP implementations employ different flow control measures to deal with network congestion. Such measures are also referred to as TCP congestion control mechanisms to differentiate them from TCP congestion avoidance mechanisms that attempt to prevent congestion from occurring. A series of TCP implementations, developed at the University of California at Berkley, are among the most widely installed and used. The Berkley TCP has undergone changes in many areas, especially flow and congestion control. The early 4.2BSD TCP contained no network congestion control measures, relying entirely on the receiver's advertised window to control flow. Even though TCP allows the receiver to control the transmitter's flow by advertising a smaller window, it does not specify how the network controls that flow in the event of congestion.

The need for explicit congestion control measures in the Berkley TCP led to the 4.3BSD and 4.3BSD$^+$ slow-start implementations. Both have adopted an explicit congestion control mechanism called the congestion control window *cwnd*. *Cwnd* is adjusted depending on the network congestion level, and the TCP transmission size is the minimum of the receiver's advertised window and *cwnd*. Network congestion is signaled through packet dropping, which the TCPs detect using retransmit timers. The two implementations differ in their congestion

handling and avoidance measures, such as the calculation of the retransmit timeout period, and the adjustment of *cwnd* upon congestion. The latest Berkley 4.3BSD$^+$ slow-start TCP included many improvements over previous implementations [Jac88].

The 4.3BSD$^+$ slow-start TCP is considered a breakthrough in congestion control and is expected to perform significantly better than previous implementations. Its new retransmit timer algorithm, which incorporates round-trip time variance estimates and exponential backoff, has been shown both robust [MT89] and efficient [Jac88]. Another major improvement in this implementation was the congestion control and avoidance algorithms. Upon congestion detection, Slow Start shuts the send window to *control* congestion and initiates a quick but gradual flow increase (slow start) to recover without causing big upsets to the equilibrium of the system. However, since the previous window size led to congestion, the new window is adjusted to half of the previous one to *avoid* repeating the congestion failure mode. This window adjustment may cause some valuable bottleneck bandwidth to be freed. To utilize it, the TCP slowly increases its window size to test the new network bandwidth limit. The TCP congestion control and avoidance algorithms rely on the network to signal congestion upon reaching this limit.

1.2 Congestion Control

Flow control is a distributed measure which functions on a per connection basis in response to a network congestion signal. *Congestion control*, on the other hand, is a centralized measure that affects all connections contributing to a bottleneck's congestion. The gateway, which holds the local buffers for itself as well as its output links, detects congestion by monitoring the sizes of its input and output buffers [2]. Congestion detection and signaling are basic functions of any gateway protocol; although, the approach differs from one design to another.

The choice of a gateway congestion control policy is dependent on the basic network service offered and the end-to-end protocol supported. These factors contribute to the amount of state information kept and processing overhead required. Other performance criteria that should

[2]The term *Congestion Control mechanism* was originally intended for the gateway but was later borrowed by the TCP as well. It should be preceded by 'TCP' whenever ambiguity might arise.

be observed are fairness in the allocation of resources to different connections, low resource utilization for a set of transferred data, reasonable queueing delay, and minimal probability of packet loss.

Most computer networks adopt either virtual circuits or datagrams as the basic network service. Virtual circuits (VCs) are end-to-end connections providing reliable network service below the transport layer, as in IBM's System Network Architecture (SNA) [GY82]. Like the transport connections, VCs require setup and termination, as well as state information maintenance at the intermediate gateways. Datagrams, on the other hand, are unreliable message carriers that require minimal setup and support overhead from intermediate gateways, relying on the higher transport layer to provide the necessary reliability. Datagram service is appropriate for heterogeneous environments supporting a variety of vendor products and protocols, such as the TCP/IP internetwork (Internet) [RFC791].

Gateway congestion control policies may require the maintenance of some state information for connections passing through them. The amount of state information kept should be matched by that required by the network service. VCs maintain a lot of state information, allowing the gateway protocol to have a better view of the traffic sources and better control of the congestion contributers. More control, though, comes at an expensive price of higher storage demand and setup overhead. The internet does away with the redundant reliability of the VCs for the flexibility of datagrams. Internet gateways keep little state information, providing minimal view of the congestion sources. Any state information required by the congestion control policy is an overhead to the Internet gateway protocol.

Computer networks may be designed to support certain transport protocols with specific gateway congestion control requirements. The amount of gateway support and its required overhead depend on the end-to-end protocols and their flow control schemes. In SNA, the transport connection adjusts its transmission window size depending on the level of congestion flagged by the gateways, in packets flowing back to the sender. The TCP flow control mechanism, on the other hand, requires minimal network support relying on gateway packet drop as a congestion indication. Flexibility in the gateway design, low processing and storage overhead, and robustness are advantages of the TCP approach. Inaccurate congestion signaling by the

gateway, though, could result in unfair distribution of bottleneck capacity, as well as lowered overall performance.

1.3 Gateway Congestion Control Policies

To achieve good performance, the Internet requires a gateway congestion control policy that can support and complement the flow control function of the TCP. Several policies have been proposed and tested, either in real or simulated network gateways. [IETF89] describes and assesses some of them, including Source Quench, Fair Queueing, No Gateway Policy, and Congestion Indication. The shortcomings and disadvantages associated with each of these policies have led the IETF Performance and Congestion Control Working Group to propose a new gateway congestion control policy, Random Drop.

Most Internet gateways do not employ any congestion control policy. They control traffic transients by buffering packets arriving while the gateway is busy. The maximum-size transient that might be accommodated depends on the gateway buffering capacity. Once this capacity is exceeded, the gateway turns to discarding packets as a way of controlling local congestion and hopefully slowing down the responsible TCPs. As long as the buffers are full, it will continue to discard packets in the order they arrive. No policy is used to differentiate between the different connections and identify those that are using more than their fair share of the bandwidth. Such explicit policy is actually costly to implement in Internet gateways. As explained above, the gateways support the datagram network service, which requires minimal connection information maintenance.

Random Drop is a compromise between the congestion control policies supported by the VC and datagram approaches. It does not require any additional state information; nontheless, it possesses an inherent ability to distinguish among the different connections using the gateway. Random Drop is intended to give feedback to users whose traffic congests the gateway, by dropping packets on a statistical basis. Buffered packets awaiting service have equal probability of being dropped, upon detecting congestion. Random Drop relies on the hypothesis "that a packet randomly selected to be dropped will belong to a particular user with probability proportional to the average rate of transmission of that user" [IETF89]. Dropping such packet

signals the appropriate user, the one currently using more than its fair share, to reduce its network load. Random Drop should be able to prevent misbehaving or aggressive connections from achieving better performance, as well as from hurting the performance of other connections. It is this ability to statistically differentiate between the different connections that contributes to the inherent fairness of Random Drop.

Early Random Drop is a variation on the simple Random Drop aimed at avoiding congestion by predicting when it will occur and taking measures to prevent it. When traffic transients become large enough to signal imminent congestion, the gateway begins to drop packets at a rate that is derived from the current network congestion level. The goal of early Random Drop is to signal those connections, whose demand is higher than their fair share, to reduce it before congestion occurs. The gateway load level at which Early Random Drop should be enacted is a function of the users' loads, the TCP flow rate, the end-to-end delay, and the gateway buffering capacity.

1.4 Thesis Plan

In [Jac88], Jacobson outlined the next research step, following the 4.3BSD$^+$ slow-start TCP, as the "the gateway side of congestion control." He described the desired goals of gateway congestion control "to control sharing and fair allocation." IETF's proposed Random Drop policy seems a logical candidate for implementation in Internet gateways, if it succeeds in achieving its design goals.

This thesis is an analysis of the performance of Random Drop as a gateway congestion control policy. The experimental studies are conducted using the MIT Network Simulator, which is described in Chapter 2. The network model underlying the experimental analysis is also outlined, along with the network topologies used to demonstrate the research results.

Chapter 3 is an investigation of congestion in network environments supporting 4.3BSD$^+$ SLOW START TCPs and gateways employing No Gateway Policy congestion control. These environments are typical of today's networks, and the simulation studies should demonstrate the shortcomings associated with present congestion control policies.

Chapter 4 introduces simple Random Drop and evaluates its performance in several network configurations. The experiments are conducted in homogeneous environments consisting of well-behaved SLOW START TCP connections, as well as heterogeneous environments containing more aggressive TCP connections. The effectiveness of Random Drop in controlling congestion fairly and efficiently in the presence of malicious TCP implementations is a major aspect of the congestion control policy. The results of these experiments shall be compared to those obtained in Chapter 3 to assess the improvements expected of Random Drop over the current No Gateway Policy congestion control.

Early Random Drop is discussed in Chapter 5. The functionality and objectives of the policy are described and a A crude implementation is provided to demonstrate the important aspects of Random Drop. This chapter does not actually attempt to develop an efficient implementation of the policy but rather to show its promising strengths, laying some grounds for further research on the topic.

The thesis results are summarized in Chapter 6. The shortcomings associated with Random Drop will be highlighted to explain why the policy fails to achieve the proposed goals. The datagram network service underlying these congestion control policies is reexamined and suggestions are made for more intelligent network services and corresponding congestion control protocols.

Chapter 2

The Simulation

2.1 Network Model

2.1.1 The Network

A computer network consists of an interconnected set of communication and switching elements. The switches, also known as gateways, route packets to their designated destinations by switching them from the input links, on which they arrived, to their appropriate output links. Each gateway is equipped with input and output buffers to accommodate transient traffic and prevent it from congesting the gateway or its output links.

Since Random Drop was proposed as a quick solution to the congestion problems that the Internet is likely to face in the near future, it is only reasonable to adopt a network model that reflects this intention. The Internet consists of a collection of local and wide area networks, that are interconnected through gateways. Local area networks use fast multi access links to provide communication over small geographic areas; while, wide area networks are composed mostly of slow point-to-point links that can provide national and cross-continental communication. Because they are the slowest network components, point-to-point links can lead to network congestion whenever the demand on them exceeds their capacity. The heavy demand manifests itself as long queues at gateway output buffers corresponding to the bottleneck links. In the worst case, the continually increasing demand exhausts the buffers and leads to network congestion.

The network model employed here will emulate these aspects of the Internet, with strong emphasis on network environments susceptible to congestion problems. The network service underlying the model is an unreliable datagram service with minimal state information maintained at the gateways. The network attempts to deliver user packets but without any guarantees, relying on the transport layer to recover from resulting errors or failures.

2.1.2 The Connection

The network model described will support end-to-end communication through transport level TCP connections. In principle, this includes all implementations conforming to the TCP specifications. Nonetheless, the experimental analysis conducted below will emphasize the Berkley 4.3BSD$^+$ SLOW START implementation because of its highly developed and used flow control algorithms, which directly influence current Internet congestion problems.

2.1.3 The User

The term user will generally refer to any entity communicating over the network through a TCP connection. Although a bulk-data transfer model of the user will be used to simplify the experimental analysis, the research results should apply to any user.

2.2 The Simulator

The research experiments are conducted using the MIT Event-Driven Network Simulator. This is a general simulation tool that allows the implementation of any integrated set of components that interact through the exchange of events. The set of components used are users, TCPs, hosts, switches, ethernets, and point-to-point links.

The MIT Network Simulator models a dynamic network environment through the implementation of dynamic TCP connections. TCP connections are configured to have fixed users and routes. Nonetheless, their opening times and data-burst sizes may be chosen dynamically using a random number generator. As in real networks, the range of the data-burst size and the time between consecutive bursts are parameters of the sending user.

2.2.1 USER

A USER pair simulates logical communication through a TCP connection. The data burst size may be fixed or randomly generated with random spacing between consecutive bursts. The USER's burst size and frequency ranges together control the network dynamics. A USER can also have an infinite burst size simulating bulk-data transmission, as with File Transfer Protocol (FTP) users.

2.2.2 TCP

A TCP component pair simulates the two ends of a TCP connection, each of which is associated with a USER. The TCP is a simplified implementation of the actual TCP specification which simulates only the data transfer state of a connection. It can emulate the Berkley 4.2BSD, 4.3BSD, or 4.3BSD$^+$ implementations, as well as a hybrid version of 4.2BSD and 4.3BSD$^+$, that will be described later.

2.2.3 HOST

A HOST simulates the physical component on which a TCP may reside. Its input and output buffers help sustain traffic transients destined to itself or its output links, minimizing network congestion. The buffers sizes might be finite or infinite, simulating very large buffers. Besides the queueing delays that packets incur awaiting service, the gateway contributes additional transmission delays, both on per-packet and per-byte basis.

2.2.4 SWITCH

A SWITCH component is similar to a HOST, except that it may only connect links together, not a TCP and a link.

2.2.5 PPLINK

A PPLINK is a point-to-point link. Associated with it are transmission and propagation delays, as well as an error probability used to simulate data corruption.

2.2.6 ETLINK

An ETLINK is an ethernet, characterized by short transmission delay and negligible propagation delay.

2.3 Performance Metrics and Analysis Tools

The experimental analysis conducted here relies on different performance measures and analysis tools to demonstrate its conclusions. Each simulation component is equipped with status indicators of its internal operation and performance. These indicators allow the assessment of performance from both the network and end-user frames of reference.

The TCP has a partial view of the network, that is, each TCP connection knows only about its own behavior and nothing about the others. Nonetheless, a TCP has enough information about itself to assess many aspects of its own performance. The simulation TCP components keep track of their throughputs, retransmission percentages, and end-to-end delays. These distributed parameters are important metrics for analyzing the interaction of the TCP and network congestion control schemes and evaluating their performance.

The TCP frame of reference, though, has a limited view of the network and usually has to make intelligent guesses about its current state. The network components, on the other hand, have a complete local picture of all network traffic. Thus, they are capable of collecting performance data about all connections concurrently and comparing it. Of course, this is much easier in the simulator than in real networks, since it is not restricted to the limited capabilities of actual network components. For example, one of the most important performance metrics here is the bandwidth distribution of a bottleneck network component among connections sharing it. A simulation component can collect and maintain data about the different connections that allows it to calculate its bandwidth distribution, whereas an analogous real component currently keeps insufficient, minimal state information.

The performance data gathered by the simulation network components can be collective or per connection. For example, both gateways and point-to-point links calculate their overall bandwidth utilizations, buffer utilizations, and packet drop percentages. They also calculate

these metrics for each connection using them, which provides sufficient data to evaluate the distribution of the capacity and assess the fairness of its allocation among users.

The simulator is equipped with graphical analysis tools to display gathered data over time. Performance data about several connections sharing a resource can be represented on one graph using colors to differentiate them[1]. Figure 4-10, for example, shows the bandwidth distribution among six connections sharing one link[2]. Similar graphs illustrating buffer distribution and packet drop percentages are shown in figures 4-9 and 5-2. It is also possible to illustrate the flow patterns of the distributed TCPs by graphing their packet sequence numbers over time. Figure 4-7 depicts such a graph for a 4.3BSD$^+$ TCP that SLOW STARTs after packet loss.

2.4 Network Topologies

The simulation experiments conducted below require a variety of network environments to demonstrate their conclusions. Two of those topologies are general enough to be used to illustrate several thesis results and thus are described here for ease of reference. The experiments describe the network environments in further detail and justify the topologies chosen.

Figure 2-1 illustrates a fairly symmetrical network topology supporting 10 TCP connections. This is not a realistic topology but the symmetry will prove very useful in comparing the bandwidth utilizations achieved by equal demand users, which is central to assessing the fairness of Random Drop. The *US-pplink* represents a slow cross-continental link with high propagation delay. It is the bottleneck of the network, the one whose capacity is heavily contended for by all connections.

Figure 2-2 models a more realistic long-haul network supporting 14 TCP connections. There is one main long route along which most connections communicate. There are also a few connections with shorter paths crossing the main route, simulating cross traffic that is characteristic of

[1]Due to the limited number of colors available to the printer, different connections might sometimes have to be represented using the same color. Fortunately, this does not obscure the illustrated results.

[2]The bandwidth distribution is calculated over consecutive fixed time periods. New values may be merged with old values using a smoothing function to illustrate long-term performance. For smoothed and average bandwidth graphs, numerical figures of the bandwidth are also included along with the graph's color keys.

real gateways. Due to the varying paths of the connections, the network has several bottlenecks corresponding to central point-to-point links that are receiving heavy traffic.

The transmitting and receiving ends of each TCP connection and its associated USER and HOST are labeled 'a' and 'b', respectively. The connections are configured for one-way data transmission to simulate the nature of bulk-data transfer chosen to model the users.

The network components have self-explanatory names and their speeds are characteristic of real Internet components. Point-to-point links and ethernets run at typical transmission speeds of 56Kbits/sec and 1Mbit/sec, respectively; whereas, switching occurs at reasonable delays of 1ms per packet plus 1ms for each byte. Other component parameters are chosen appropriately to achieve the desired network load.

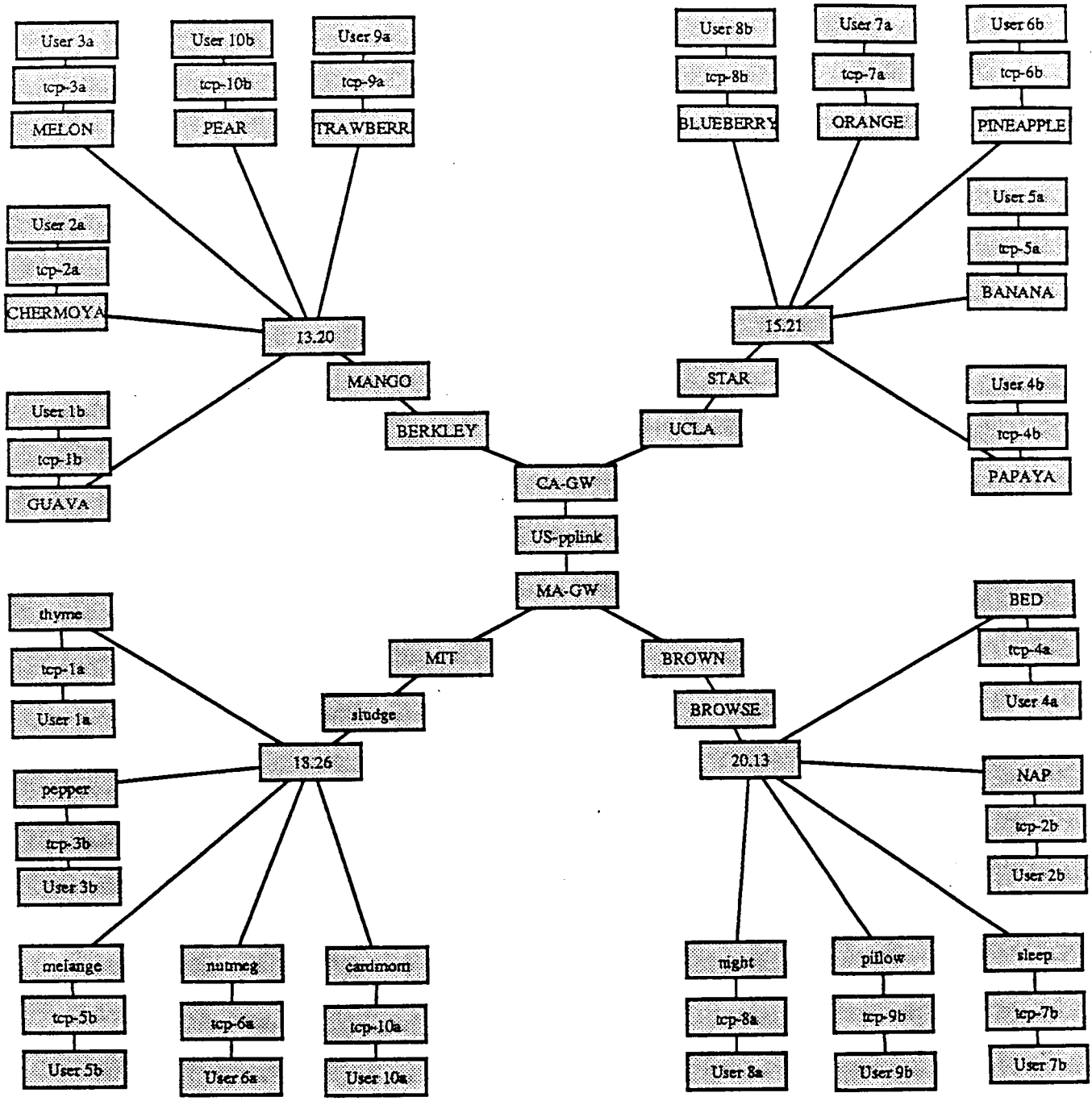


Figure 2-1: Network Topology I

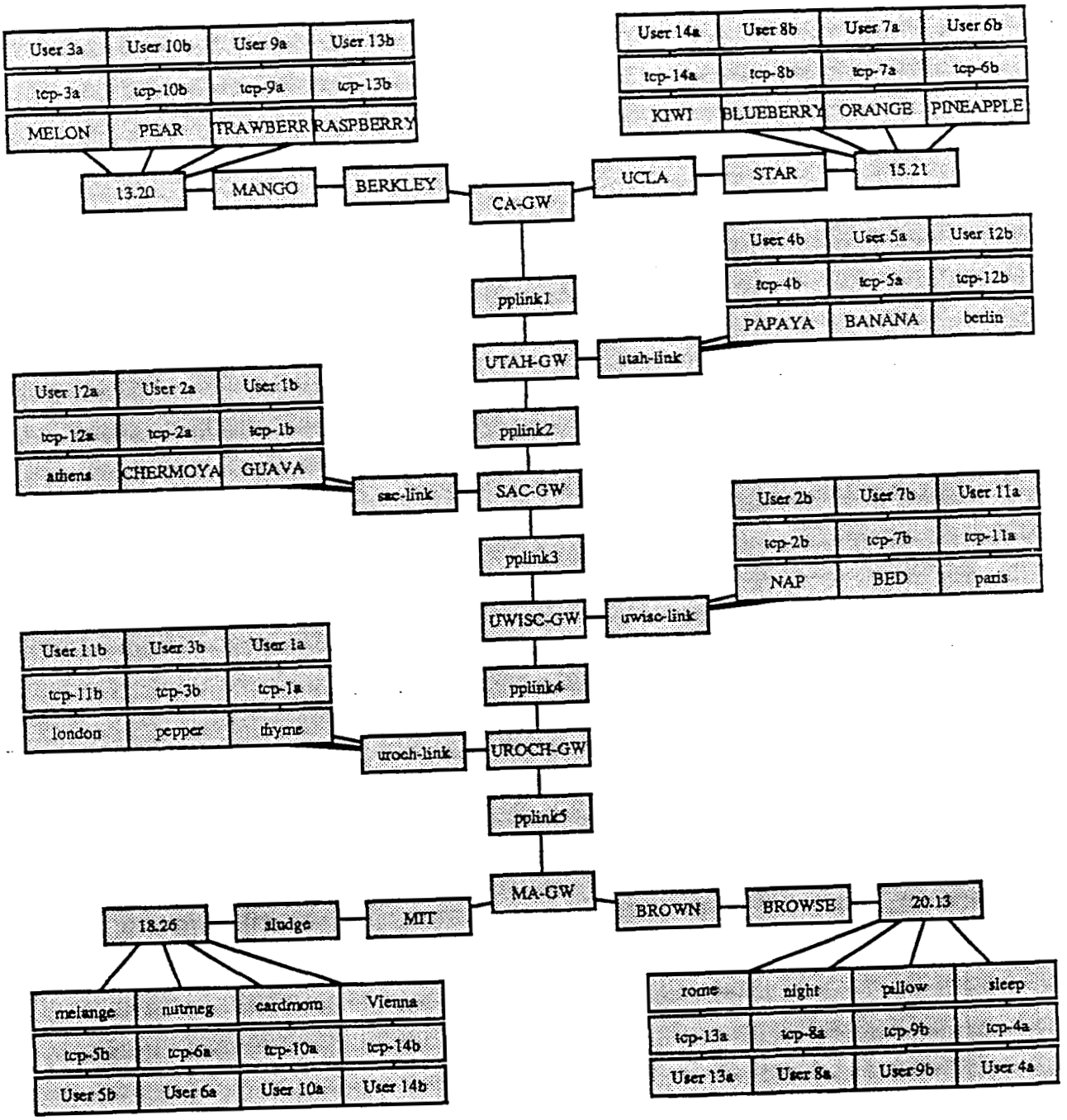


Figure 2-2: Network Topology II

Chapter 3

Network Congestion

Computer networks are very dynamic environments supporting user traffic of different characteristics and demands. The strong dynamics make it very difficult to predict the maximum network load and allocate enough capacity to service it. Instead, the network resources are equipped with enough capacity to handle reasonable traffic loads, along with congestion control mechanisms to protect themselves from excessive user traffic. The growing demand on computer networks has increased the importance of efficient congestion control schemes, that react quickly to sudden changes in traffic characteristics. The SLOW START flow control mechanism implemented in the 4.3BSD$^+$ TCP was such an effort to enact quick but controlled TCP congestion control and recovery. This chapter discusses the interaction of the SLOW START TCP with the present IP gateway congestion control scheme, that employs no dropping policy. The objective is to analyze their behavior in heavily utilized environments that are susceptible to congestion and demonstrate the phenomena contributing to it.

Network resources have limited capacities which may be exceeded by user demand. In such cases, the resources can accommodate finite traffic transients in their local buffers until they are processed. If the transients last long enough, though, they will exhaust the bottleneck's buffers causing them to overflow and the resource to get congested. Thus, network congestion is caused by the aggregation of user traffic (data packets) awaiting the service of one or more bottleneck resources. The study of this aggregation or clustering effect is fundamental to understanding the nature of network congestion.

In the TCP/IP networks, the clustering effect is evident as several different phenomena that may occur separately or concurrently. Packet clusters may form among packets belonging to the same TCP connection or packets from different connections. To distinguish between them, the first will be called Local Packet Clustering (LPC) and the second Global Packet Clustering (GPC). The LPC phenomenon is a consequence of the TCP transmission strategy used to regulate the flow of packets within an individual window. Windows provide little control over the flow rate of individual packets, allowing them to cluster. Even the additional control exercised by the latest Berkley SLOW START transmission strategy is not sufficient to prevent LPC.

More substantial packet clustering arises as an aggregation of packets of many connections at a bottleneck resource. This Global Packet Clustering is observed as two separate phenomena caused by different factors. Lossless GPC occurs in network environments with bottlenecks characterized by high processing delays but few packet drops. Such environments rely heavily on network buffering to increase throughput, even at the expense of high queueing delay. Slow bottleneck resources cause packets to queue up awaiting service. The long queues encourage other packets to join them, reducing their inter-packet distances and clustering them together. This effect usually involves most connections using the bottleneck, which causes their flows to synchronize. Since this occurs during each round-trip around the network and through the bottleneck, the synchronization will reoccur each time, causing the traffic to oscillate between being queued at the bottleneck and being transmitted by the TCPs. The significance of Lossless GPC depends on the processing delays and corresponding loads of network resources.

Another type of GPC is usually seen in networks characterized by small buffering and susceptible to heavy packet losses. This Lossy Global Packet Clustering behavior results when the demand on a resource exceeds its capacity causing incoming packets to queue up awaiting service. If such traffic transient lasts long enough, it will overflow the available buffers and congest the resource. This GPC effect should not persist because the resource will signal the TCPs to slow down and control their flow. Due to the dynamic nature of networks, though, the TCPs are designed to test the current network load limit by slowly increasing their flow back up until some congested resource tells them to slow down again. Thus, the TCP oscillates

between flow control to relieve congestion and flow maximization to improve throughput. The resulting effect is that the global packet clusters will continue as long as the total user demand exceeds the capacity of some shared resource. Generally, Lossy GPC has a more pronounced effect than other clustering phenomena, because it usually involves many connections using the bottleneck resource, subjecting them repeatedly to packet losses and performance penalities.

3.1 Local Packet Clustering

The aggregation of packets from one TCP connection in the buffers of a bottleneck resource is termed Local Packet Clustering. LPC is very closely related to the TCP transmission strategy, which is implementation dependent. Unlike windows, which are a TCP specification, transmission strategies vary from one TCP implementation to another depending on the designer's philosophy. Some of the widely used transmission strategies are those implemented by the Berkley TCPs, particularly the latest SLOW START. SLOW START is considered a breakthrough in flow and congestion control with several improvements over previous transmission strategies. Nonetheless, SLOW START suffers from a problem familiar to many other strategies, little control over individual packet flow within a window. This shortcoming of SLOW START contributes to an LPC effect in networks employing it to control end-to-end data flow.

Fortunately, the network resources impose delays on the service of packets, which effectively disperse clustered packets after a few windows. This dispersion effect helps SLOW START TCPs overcome LPC, softening its long-term effect on the network.

3.1.1 Network Delays

As packets travel through the network, they experience queueing, transmission, and propagation delays. Bottleneck resources have two opposite effects on the packet flow characteristics, one associated with incoming packets and one with outgoing packets. Packets arriving at a busy resource join its input buffer queue. The larger the queue, the higher the probability of a packet arriving while the resource is busy and joining the queue. Thus, the queueing delay reduces interpacket separation, encouraging packet clustering at the input buffers.

On the other hand, processing delays at the resource (i.e. transmission delays) insure at least

a 1-packet processing delay separation between consecutive outgoing packets. This interpacket separation results in a dispersion effect that spreads apart any packets previously clustered at the input buffer of the resource. The combined effect of the queueing and processing delays is that the resource can not usually maintain the original interpacket distance, but it can guarantee at least a 1-packet processing delay separation.

3.1.2 SLOW START Transmission Strategy

The TCP SLOW START transmission strategy has been adopted because of its effectiveness in reducing congestion by gradually merging a newly opened connection or a retransmitting connection with the current network traffic [Jac88]. Previous Berkley TCP transmission strategies, such as 4.2BSD and 4.3BSD, may transmit full-size windows all at once upsetting the equilibrium of the network and congesting the bottleneck resources. SLOW START opens the TCP sending window exponentially up to a certain threshold, then linearly until it reaches its maximum limit. The exponential sizing is adopted for the TCP to quickly reach equilibrium, followed by the linear sizing for slow probing of the current network load limit. The threshold separating the exponential and linear sizing regions is dynamically adjusted depending on the network congestion level. Upon detecting buffer overflow, the TCP deduces that some bottleneck resource could not handle its current network load (window size) and that it should slow down. Since the resource was able to handle half of the current window size, the threshold is readjusted to that safe limit. Beyond this threshold, the TCP will advance slowly (linear sizing) avoiding congesting the bottleneck resource again.

Exponential window sizing is achieved by increasing the window size by one packet for each acknowledgement received, assuming each data packet generates one acknowledgement. Thus for each acknowledgement received, the window is opened by two packets, one for the acknowledged packet and one for the new increment. If user data is always available for transmission, the TCP connection will immediately proceed to send a burst of two packets. Even though the individual packets of a window get dispersed by the delays of network resources, these small bursts can fill the gaps between consecutive packets in the following window, causing them to cluster at the next bottleneck. This LPC effect lasts until exponential sizing ceases, allowing the network dispersion effect to take over and restore smooth TCP data flow.

3.1.3 LPC Simulation Results

Because the LPC effect is local, that is it affects each flow separately, it does not pose any big threats on its own. Nonetheless, it is important to realize its existence, as it might prove significant when coupled with other network phenomena, such as Global Packet Clustering that will be described in the next section.

LPC will be illustrated by observing the data flow pattern of a typical TCP connection while SLOW STARTing. For this purpose, topology II of Figure 2-2 provides a realistic example of a long-haul TCP/IP network with several intersecting flows. Figure 3-1 focuses on the transmission pattern of *tcp-2a* during a typical user data burst. The SLOW START TCP begins transmission by sizing up its window exponentially until the maximum size is reached. The exponential window size adjustment is illustrated by the small 2-pkt bursts described above. Each window transmission consists of a string of 2-pkt bursts that grows exponentially in size yielding aggregates of 1, 2, 4, 8, and 16 packets [1]. These aggregates correspond to the Local Packet Clusters that form at the inputs of bottleneck resources during the exponential window sizing phase of TCP flow control.

After the TCP window has reached its maximum size of 1600 bytes, its transmission pattern seems to take on a new character. The packet flow becomes almost regular with nearly equal inter-packet spacing. The dispersion of packets is due to the processing delays of network resources. It is less apparent at the beginning of the transmission, because the effect is obscured by the 2-pkt bursts that form the exponential window sizing. Once the SLOW START phase is over, the Local Packet Clustering effect seizes giving way to the network dispersion effect to smooth out the traffic fluctuations. Thus, the processing delays of network resources work to regulate the TCP flow acting as an embedded rate control mechanism.

The significance of the LPC effect depends on the dynamics of the network and the availability of buffers to accommodate traffic transients at bottleneck resources. Those two factors determine how often the TCP needs to activate its SLOW START mechanism which results in

[1] In this experiment, the TCPs were configured to have maximum window sizes of 1600 bytes and packet sizes of 100 bytes.

Local Packet Clustering. Only the effect of the first factor is explored here because it is easier to isolate and illustrate. The effect of buffer size on congestion will be illustrated in the next section.

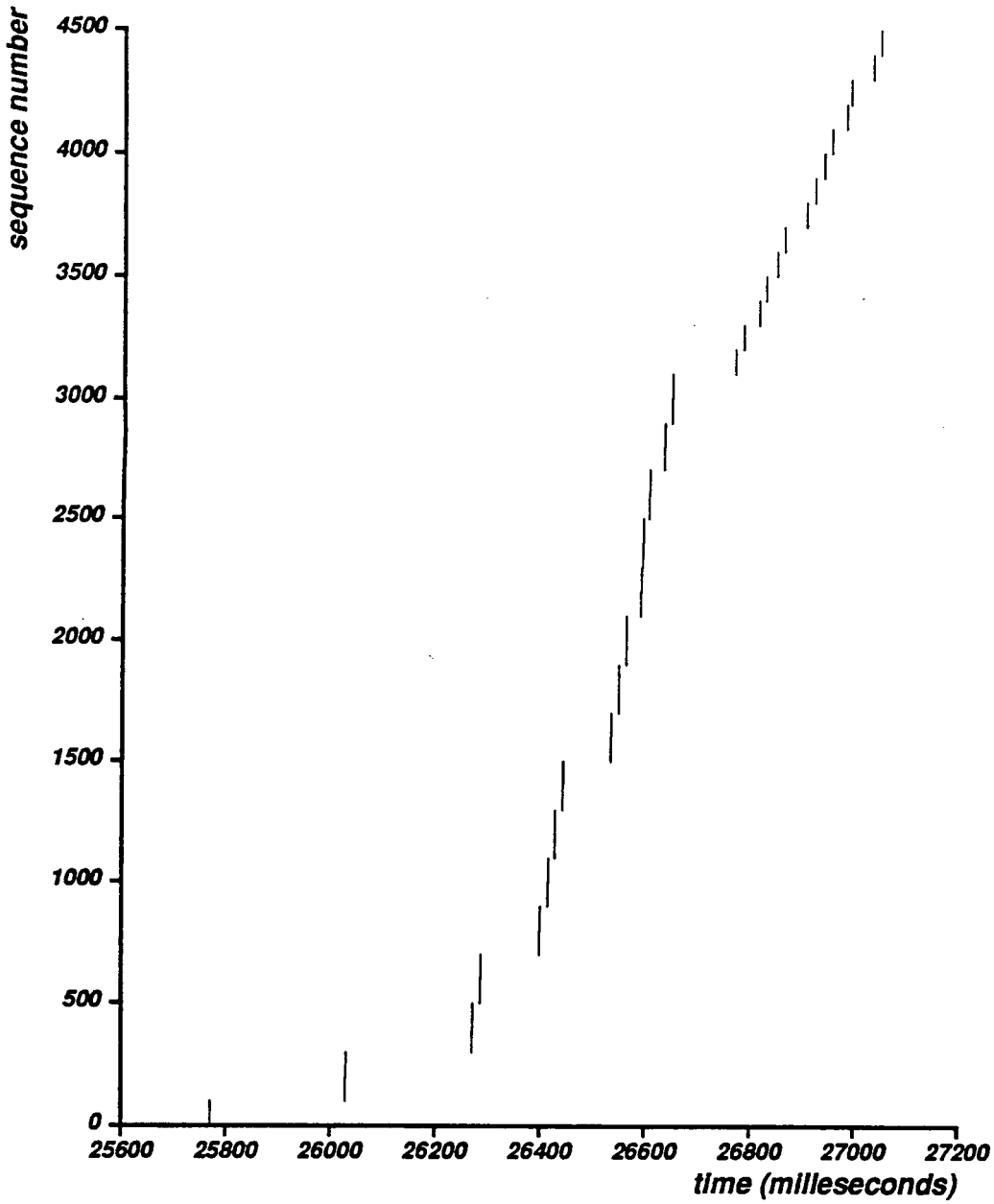


Figure 3-1: Sender sequence number logging for *tcp-2a*

3.2 Lossless Global Packet Clustering

3.2.1 Network Delays

In the previous section, the effects of delays imposed by network resources on packet flow characteristics were described. Processing delays were shown to have a positive dispersion effect on outgoing packets, which counteracts the effect of Local Packet Clustering. Unfortunately, processing delays have another negative influence on incoming packets, which could prove significant under the right conditions.

If a network resource has high processing delay and is receiving heavy traffic, then packets are likely to queue up frequently at its input. The more packets are queued up, the higher is the probability of packets arriving while the queue is busy and joining it. Thus, long queues attract packets to joining them, clustering them together and destroying their original inter-packet separation. Moreover, this effect feeds upon itself since the longer the queue, the more packets it attracts, and the longer it becomes. This causes each connection to oscillate between idleness, while its packets are queued up at some bottleneck, and near full utilization, when all the acknowledgements arrive allowing the transmission of a whole new data window. Since this effect is global, involving most connections using a bottleneck resource, it works to synchronize the flow patterns of all connections involved. Furthermore, it continues to resynchronize them after each round-trip, effectively aligning their flows to yield network-wide oscillations.

3.2.2 Lossless GPC Simulation Results

Topology II of Figure 2-2 is used to demonstrate Lossless GPC. Network resources are equipped with infinite buffering capacity simulating very large buffers. The 56 Kbit/sec point-to-point links represent the network bottlenecks with high processing (transmission) delays. Figure 3-2 illustrates the queue length for packets in the output buffers of *SAC-GW* awaiting transmission on *pplink3*. With a 16-pkt window per connection and a total of 12 connections, the oscillation peaks of 45-60 packet amplitude demonstrate an aggregation of packets belonging to several flows. Moreover this packet clustering repeats regularly and indefinitely supporting a

synchronization of network flows over long time periods [2].

The significance of the Lossless GPC effect depends on the processing delay of the bottleneck resource and the total demand on it. These parameters determine the amplitude of the resulting oscillations. Moreover, Lossless GPC is supported by the failure of TCP to exercise rate control over the flow of its individual packets. Thus, a packet cluster that develops in the buffers of some bottleneck resource will produce a cluster of acknowledgements generated by the receiver, followed by a cluster of new data packets generated by the sender. Current TCP/IP networks lack an explicit mechanism that would work to break such vicious packet clusters as they travel indefinitely around the TCP connection loop.

[2] All the connections in this experiment have opened at different times during the first half second and continued transmitting indefinitely with maximum round-trip delays of 1.5 second.

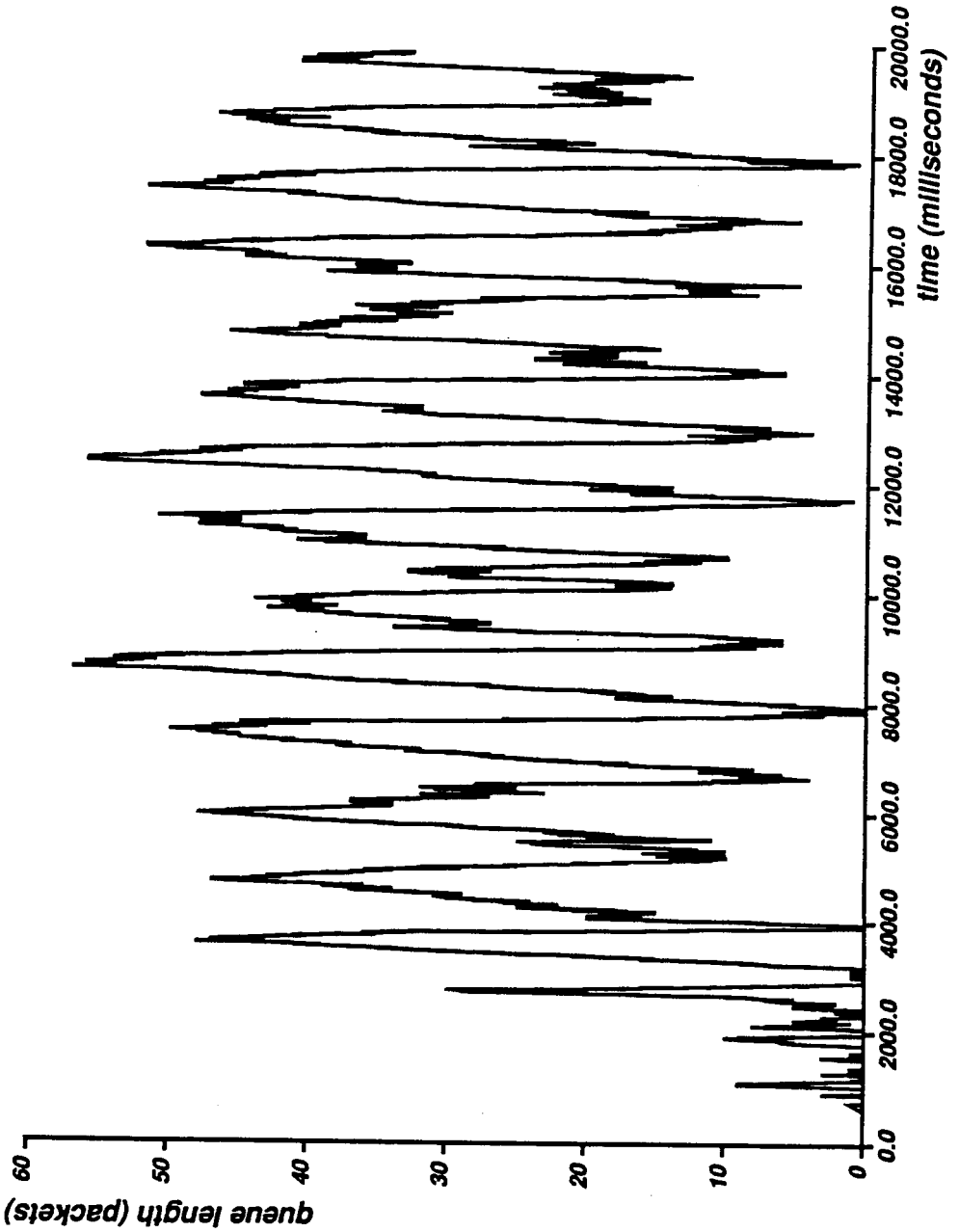


Figure 3-2: Output queue length for *SAC-GW* in the *pplink3* direction

3.3 Lossy Global Packet Clustering

Another Global Clustering effect occurs in network environments characterized by limited buffering and correspondingly reasonable queueing delays. Lossy Global Packet Clustering (GPC) is a product of the interaction of the gateway and TCP congestion and flow control mechanisms in heavily utilized network environments. These mechanisms are designed to control network congestion while maintaining a high utilization of the resources. Pursuing a balance between these two conflicting objectives, though, can lead the network into an oscillatory behavior between congestion recovery and utilization optimization. This behavior translates itself into oscillations in the load of bottleneck resources, seen through their input queue sizes. Moreover, the gateway congestion control policy seems to have a synchronizing effect on the transmission patterns of TCP connections, which leads to synchronization of the connections LPC effects. This works to increase the burstiness of the load on the bottleneck resources making things even worse.

3.3.1 Gateway Congestion Control

The gateway congestion control mechanism is a combination of two schemes, local buffering and packet discarding. Heavy transient traffic is controlled through buffering packets arriving while the resource is busy. The maximum-size traffic transient that might be handled through buffering depends on the size of the gateway buffering capacity. Once this capacity is exceeded, the gateway turns to discarding packets, as a way of controlling local congestion and hopefully slowing the TCPs down. As long as the gateway's buffers are full, it will continue to discard packets in the order they arrive. In doing so, the gateway is relying on the TCPs to detect congestion and take measures to control their flow and recover from any losses they might have incurred.

3.3.2 TCP Congestion Avoidance

The TCP has been designed to detect network congestion through the use of retransmission timers. If the TCP does not receive an acknowledgement for a packet after a certain time of its transmission, it assumes that the packet was discarded by some congested resource on its

path. Upon detecting this loss, the TCP activates its congestion control mechanism, that is exponential window sizing, to reduce network congestion and recover from its losses. Once congestion is controlled, the TCP attempts to optimize its performance by maximizing the throughput while avoiding congestion. Congestion avoidance is achieved through linear window sizing which slowly increases the throughput until the TCP load reaches its limit. The TCP traffic load is restricted statically by its window boundary and dynamically by the current network load. When one of the congested gateways, on the path of a TCP connection, exhausts its buffers and begins to discard packets, then the network imposes a dynamic limit on the connection's load. This load is a function of the number of connections using the congested gateway and their aggregate network load. Linear window sizing optimizes performance by increasing the TCP throughput until the static or dynamic limit is reached.

3.3.3 High Network Demand

The capacity of a network is limited by the capacity of its resources. When the demand on the network exceeds the capacity of some bottleneck resource, packets from the connections passing through that resource aggregate at its input buffers. This Global Packet Clustering effect, caused by the high network demand, is inevitable in a dynamic network environment, and the gateway and TCP are equipped with effective measures to control it quickly.

The congestion control mechanisms used in the TCP/IP networks often provide powerful control of congestion. Unfortunately, their effect seems to be short lasting. In a heavily utilized environment, heavy network load often leads to congestion. Even though the TCP and gateway succeed in relieving it and recovering from the loss, they are not designed to remember any information about the congestion failure. Following each congestion recovery, the TCP slowly probes the network load limit until it reaches it, forcing the network into another congestion failure. Thus, the interaction of the gateway congestion control and TCP congestion avoidance mechanisms causes the TCP to oscillate between congestion recovery and performance optimization.

3.3.4 Lossy GPC Simulation Results

Topology II will be used again to demonstrate Lossy GPC. In this experiment, though, network resources are equipped with finite buffering capacity, which is frequently exceeded by the heavy user demand. Figure 3-3 shows the buffer distribution for *pplink3* in the *SAC-GW - UWISC-GW* direction. *Pplink3*'s service queue seems to fluctuate between full buffer utilization and idleness quite regularly. Considering the fact that user demand is constant over the entire graph time period, these fluctuations support the hypothesis of oscillatory TCP flow pattern. Such a typical pattern is illustrated in Figure 3-4. The oscillatory TCP flow character exerts uneven load on network bottlenecks encouraging packet clustering. Moreover, each SLOW START, shown in the figure, is accompanied by an LPC effect which only increases the load on the bottlenecks and any potential clustering in their input buffers.

Besides the oscillatory character of each TCP data flow, the large-amplitude oscillations of *pplink3*'s queue support some synchronization of these flows. Just as in the previous section, bottleneck queueing delays have led to the synchronization of the data flows of several TCP connections, the gateway congestion control mechanism seems to induce a similar effect. Whenever a congestion failure occurs, several connections are hurt in the congestion control process. The overlap of data flows from several connections coupled with the first-come-first-drop gateway congestion control mechanism will typically cause several connections to lose. All connections hit during the drop will activate a SLOW START following the retransmission timeout, in the process aligning their data flows. It is this synchronization of several flows that is responsible for the large oscillations and significance of the Lossy GPC effect. The application of no drop policy by the gateway congestion control mechanism causes connections to be penalized blindly without concern to who is really contributing to the bottleneck's overload.

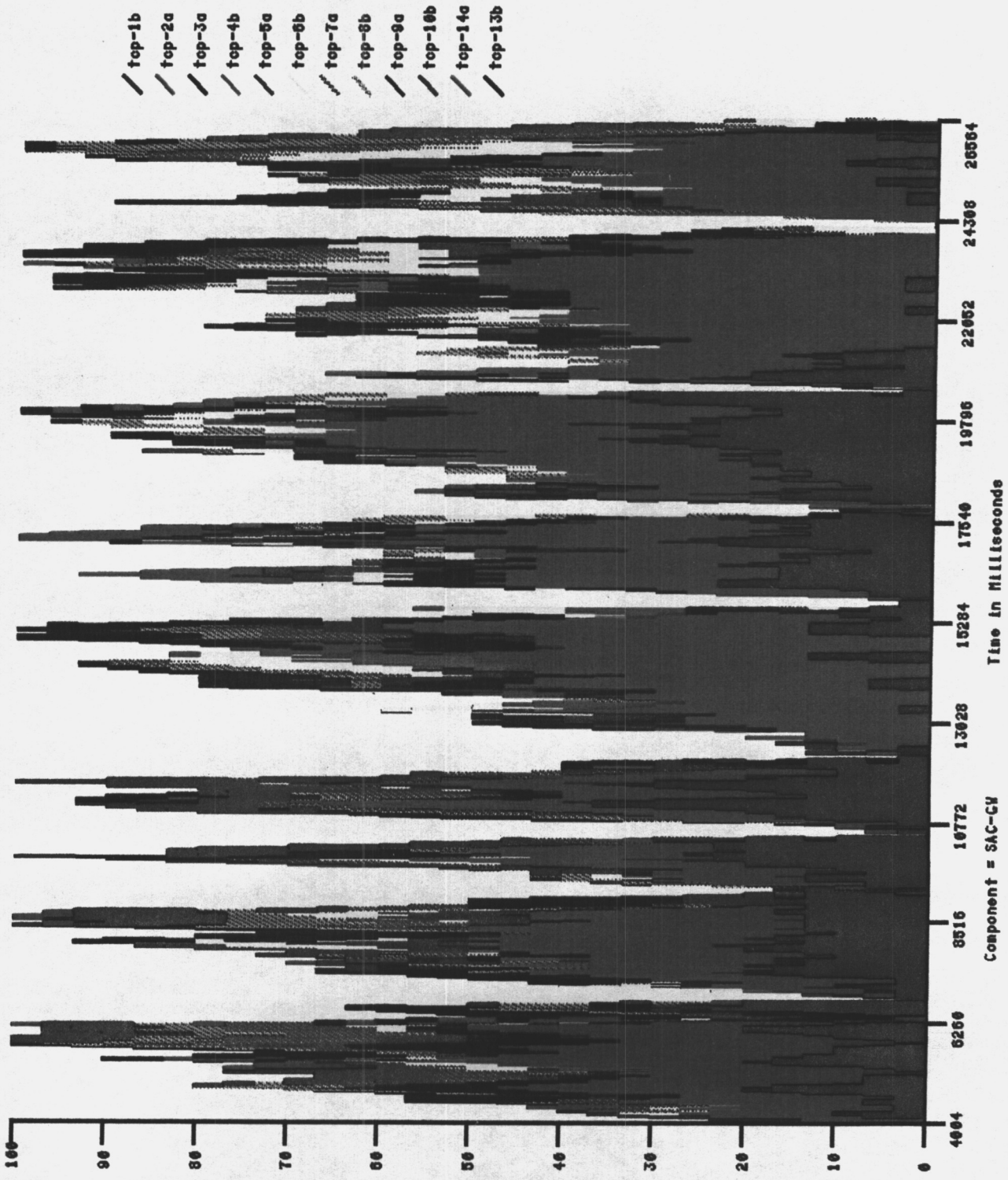


Figure 3-3: Buffer distribution for *pplink3* in the *SAC-GW - UWISC-GW* direction

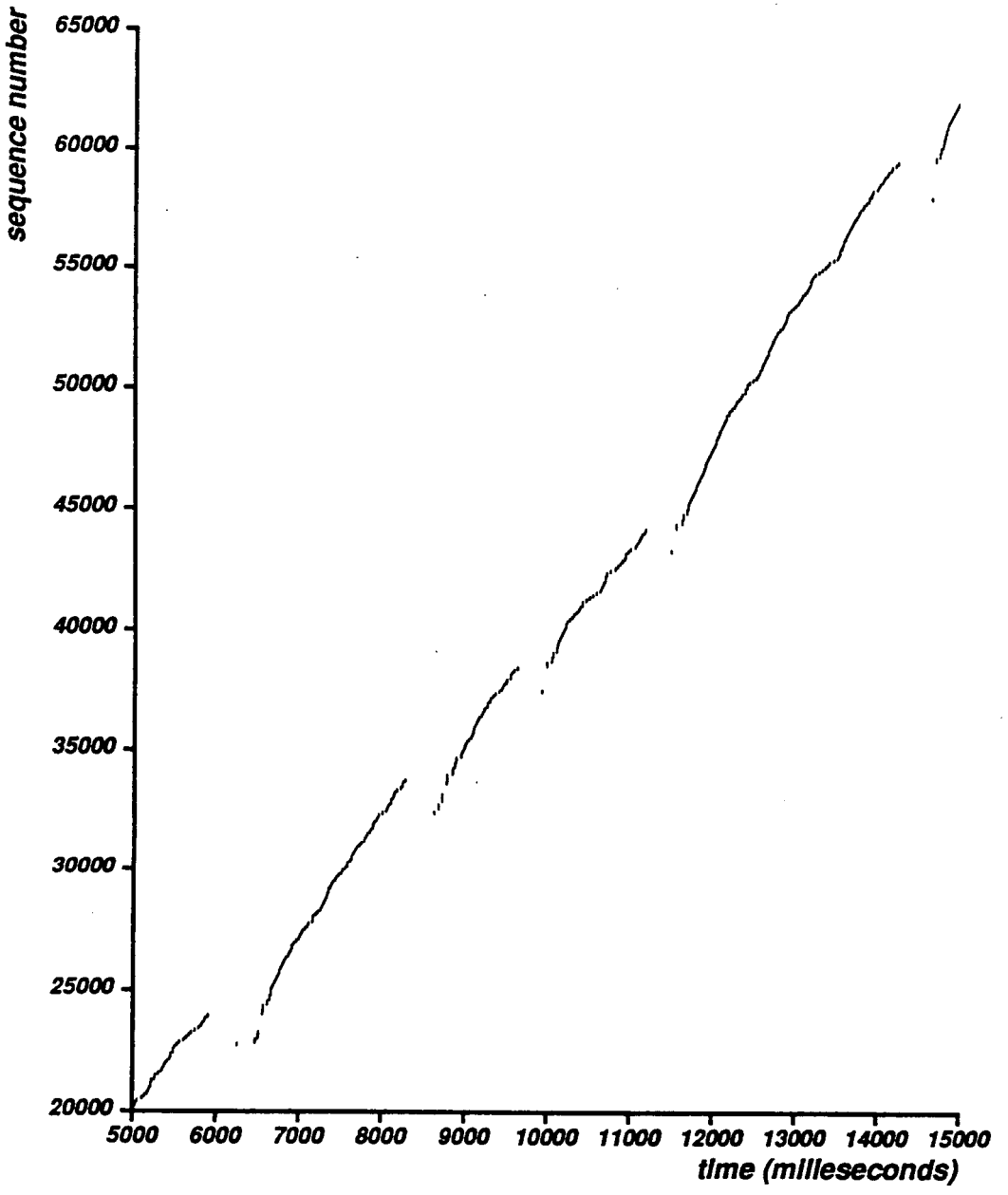


Figure 3-4: Sender sequence number logging for *tcp-2a*

Chapter 4

Random Drop

The Internet is expected to face significant congestion in the next couple of years, which prompted a review of the current congestion control policies. Remote file systems and persistent name resolvers are some examples of large contributers to the Internet traffic, which are expected to consume a large portion of the available bandwidth [Man89]. Since the need is for the near future, any potential congestion control policy should be easy to add to existing Internet gateways. The appeal of the proposed Random Drop policy follows from its implementation simplicity, its conformance with the layering hierarchy and datagram network service, and its strong potential to provide the much needed congestion control.

Random Drop was proposed to provide both congestion control and avoidance functions to network gateways. This chapter focuses on the congestion control aspect of Random Drop, while congestion avoidance is discussed in Chapter 5. The Random Drop congestion control policy is intended to reduce the congestion at a bottleneck resource by identifying users that are consuming more than their fair share of the available capacity and signaling them to reduce their network load. User demand on the bottleneck capacity is identified implicitly through statistical approximation. Random Drop relies on the hypothesis that the distribution of available buffers at a bottleneck resource reflects the distribution of its capacity among the different users. That is, the percentage of buffers occupied by a particular user is proportional to the average transmission rate of that user. Therefore, a buffered packet selected randomly to be dropped will belong to a particular user with probability proportional its average transmission rate.

Users that attempt to consume more than their fair share of the capacity will also consume a larger percentage of the buffers and thus see a higher drop probability. It is this inherent ability to distinguish between the demands of the different users and signal only those that are contributing to the congestion that makes Random Drop an attractive policy.

Random Drop is also viewed as an improvement to No Gateway Policy packet dropping when the collective demand on a bottleneck resource is large enough to overflow the available buffers. When No Gateway Policy is employed, packets arriving at a congested resource see full buffers and are dropped exactly in the order they arrive. Since traffic of different users is not necessarily uniformly intertwined, the percentage of packets dropped that belong to a particular user may not reflect that user's buffer utilization. It is possible that a cluster of packets belonging to one user, one that is using less than its fair share of the capacity, arrive at a congested resource just in time to overflow its buffers. The well-behaved user ends up paying for the misbehavior of some other users. Intuitively, Random Drop should only do better than No Gateway Policy as far as distributing the available capacity fairly among the users.

Having described the inherent characteristics and goals of Random Drop, this chapter proceeds to verify its underlying assumptions and assess its performance in various network environments. The fairness of Random Drop in penalizing the congestion contributers will be the focus of the analysis. Accurate identification of the congestion contributers should also yield more efficient overall behavior. The chapter concludes with a comparison of the performance of Random Drop to that of No Gateway Policy congestion control to assess the feasibility of its incorporation in existing Internet gateways.

4.1 Performance Criteria

4.1.1 Fairness

Besides its vague qualitative meaning, fairness has several quantitative definitions [JCH84, RCJ87]. The fair allocation of resources depends largely on the users' demands. The simplest scenario is when the users have equal demands entitling them to equal shares of the capacity. In reality, though, there are different classes of users with varying network demands. Fair allocation in such situations is very tricky and difficult to provide by a mere modification of the congestion control policy. More fundamental changes to the network service model are required to provide different qualities of service as requested by the user [Zha89]. In the Internet, support for unequal shares of the resources would have to be provided by a separate protocol that relies on an equal-share network service.

Assessment of the fairness of Random Drop shall be based upon equal-share fair allocation; that is, whatever the demands of the users are, they are entitled to equal shares of the network capacity. Any excess unused capacity is free to be grabbed by any user requiring it, until it is needed by a user whose demand is below his fair share. This definition of fair allocation is compatible with that which Random Drop promises to deliver. Users are allowed to consume the resources capacity any way they wish as long as they do not exceed it. If they do, Random Drop will statistically hamper those connections whose demand exceeds their fair share capacity. The connections should then reduce their network load, returning some of the capacity back to its lawful owners. Remember that Random Drop is a statistical policy, so that this behavior manifests itself more clearly over longer time periods in which the effect of traffic transients has diminished.

4.1.2 Power

Network performance is optimized by maximizing the total throughput and minimizing the average queueing delay at bottleneck resources. Since these are two conflicting criteria, a compromise between them is often taken as the most optimal objective. Power is one such

metric that models the trade-off between throughput and delay. It is defined as

$$Power = \alpha \times \frac{throughput}{delay}$$

[1] [Kle79, IETF89]. Thus, from the network point of view, it is considered desirable to maximize the power for all network resources; nontheless, in reality this is probably impossible due to the conflicting needs of different resources.

Optimal network performance, though, does not necessarily guarantee good performance to individual TCP connections. Besides the need for fair allocation of network resources among the users, it is also important to minimize the retransmission rate of their data. Only by doing so, can the network capacity be efficiently utilized. Maximizing the resource capacity spent transmitting real user data (rather than retransmitting old data) would insure that the ends will also achieve high throughput. The low network queueing delay combined with the high user throughput should then produce the desired user power.

As explained in Chapter 3, TCP/IP networks are subject to load oscillations due to packet clustering phenomena. Needless to say, these oscillations are very undesirable, as they result in high queueing delays, high retransmission rates, and reduced throughput. Ideally, networks should have bottlenecks with average buffer utilization of 1 packet, including the one that is being processed. This would imply maximum utilization of bottleneck capacity at no queueing delay expense.

A more realistic objective, though, is to aim at keeping the resource queueing fairly short and stable. Obviously, this should reduce the queueing delay and increase the utilization of network bottlenecks. Moreover, decreasing the burstiness of network traffic should lessen the danger of buffer overflow and the resulting retransmissions.

The assessment of Random Drop's performance will be only qualitative, aiming to determine whether it has the sound assumptions and behavior needed to fulfill the performance criteria outlined. Random Drop's promise to identify and penalize only the users contributing to

[1] α is a variable that reflects the relative importance of throughput versus delay. A value of 1 is chosen if equal emphasis is placed on both criteria.

network congestion means that substantial variations in queue size result from the slowing down of aggressive users only. This would protect the good users, keeping their retransmission rates low and their throughputs high. To maintain low queueing delays, buffers of 'reasonable' size are recommended. SLOW START TCPs are designed to continuously test the limits of their networks, and they do that by dynamically increasing their window sizes until the buffers of some bottleneck overflow, signaling them to slow down. Thus, large buffers can incur high queueing delays, while small buffers result in too many retransmissions.

To summarize, the ability of Random Drop to implicitly identify and control the traffic of each network user is the most critical performance criteria. It is the one that will decide how well Random Drop will perform as far as the other criteria are concerned.

4.2 Performance Assessment

Since Random Drop was proposed, there have been arguments in its favor and arguments against it. The experiments included here do not necessarily represent a conclusive test of its performance, but they do demonstrate some of its strengths as well as some of its shortcomings. The objective of these simulation experiments is to become familiar with the nature and behavior of Random Drop, and how it fits in the current network model.

4.2.1 Benefits of Random Drop

Computer networks are event-driven systems, whose behavior is the outcome of a succession of events that activate each other in some regular fashion. A repeated sequence of events, that may result from such regularity, can cause the network to be locked into one stable pattern that might be disadvantageous either to it as a whole or to the individual connections. To counteract this negative stability, some randomness can be injected into the network to break up the regular chain of events that could lead to an unfavorable outcome. It is this aspect of Random Drop that was believed to be a strong improvement over the current No Gateway Policy congestion control. The following experiments will demonstrate the randomness effect on network flow patterns.

In all of the experiments conducted in this chapter, the users of the TCP connections were chosen to emulate FTP users, which are actually simulated by infinite demand sources. Such users impose equal demands on the network, simplifying the Random Drop analysis. Moreover, the results obtained below will make it unnecessary to test Random Drop in the presence of unequal demand users.

The network environments described by the experiments in this section require some of their flows to be able to stabilize. In order for this to occur, a TCP connection must have a static window size small enough to be accommodated within the buffers of any network resource. The intuition behind this choice will become clear as the network scenarios are described below.

4.2.1.1 Experiment 1: Identical TCPs with Different Open Times

Network environments are usually characterized by random connection establishment and clearing. In such environments, connections that start earlier and manage to stabilize their flows might be at an advantage compared to connections that are attempting to join some already started and stable set of flows. In a network characterized by scarce resources, there is a danger of some small set of flows grabbing most of the resources while the other flows fight over the leftovers.

Topology III of Figure 4-1 describes such a network scenario. It consists of two identical TCP connections competing for the capacity of a bottleneck point-to-point link, *PPLINK1*. The link has a buffering capacity that can accommodate 40 packets, while each connection has a maximum window size of 25 packets. The network end-to-end delay is short enough to cause most packets to be queued up at the bottleneck. This is where the problem arises, since *PPLINK1*'s buffers can not hold the packets of of both connections combined, the two will have to contend for the resources. Nevertheless, the connection that opens first has a head start on the contention.

With No Gateway Policy employed while dropping packets to relieve congestion, whoever arrives while the buffers are full will be penalized. Since the first connection has already started and is holding a fixed share of the available buffers, any of its packets that leaves the bottleneck will be shortly replaced by a new data packet. On the other hand, the second connection will try to open its window and stabilize, but meanwhile will force the buffers to overflow and be penalized for the congestion. This behavior is clearly illustrated in Figure 4-2 showing the buffer distribution for *PPLINK1* in the *GW1 - GW2* direction. *TCP1A*, which opens first, occupies a fixed portion of the available buffers while *TCP2A* oscillates continuously in an unsuccessful effort to stabilize. Clearly, this is not a fair treatment to two identical users with equal demands.

Figure 4-3 shows a typical buffer distribution for *PPLINK1* when Random Drop is used. Both connections are penalized for the congestion, instead of simply the unlucky one that started later. Since the two connections differ only in their opening times, Random Drop should be able to overcome this discrepancy and allow both to attain similar performance levels. In

fact, a comparison of the long-term average throughputs of the two connections supports this hypothesis and Random Drop's improvement over the current No Gateway Policy congestion control.

4.2.1.2 Experiment 2: A LAN versus a WAN

Computer networks generally interconnect heterogeneous systems with varying capacities and services. The networks are expected to conceal this heterogeneity, providing transparent reliable service to their users. Thus, a pair of users communicating over a cross-continental connection should receive a similar quality of service as a pair communicating within the same building, provided that their demands are equal. Anybody with some knowledge of communication, though, would tell you that this is nonsense. Wide area networks (WANs) are technologically much slower that local area networks (LANs)[2]. Besides their slow speeds, WANs also consist of concatenations of network components of mismatched speeds. These factors make them very susceptible to congestion. To make the problem even worse, the TCP flow and congestion control mechanisms intrinsically favor LANs with faster congestion detection and recovery.

In addition to the obvious WAN problems, there are some less obvious. One such problem arises when two TCP connections, one running over a WAN and one over a LAN, are competing for the limited capacity of a shared bottleneck. Figure 4-4 demonstrates such a scenario. The two TCPs have similar paths except that for one hop, *TCP1A* is crossing a fast ethernet while *TCP2A* is crossing a slow long delay point-to-point link. Both connections' paths pass through *PPLINK1* which is the main network bottleneck. The link is not fast enough to service both connections and its buffers can accommodate only a maximum of 30 packets. This seems sufficient for *TCP1A* whose window consists of 15 packets, most of which are always queued up at the bottleneck. The nearly fixed buffer utilization of *TCP1A* is in contrast with the highly oscillatory buffer utilization of *TCP2A* as shown in Figure 4-5. The similarity between this figure and Figure 4-2 should be apparent since they both demonstrate the same phenomenon generated under different circumstances.

Just as in the previous experiment, *TCP1A* gets a head start on the other connection,

[2]The speed comparisons apply to current large-scale computer networks and not to a few isolated fast links.

because of its short end-to-end delay, and manages to stabilize and seize a fixed portion of the available buffers for its entire duration. Meanwhile, *TCP2A* is struggling to open up its window, repeatedly overflowing the bottleneck's buffers in the process. Because of its large end-to-end delay, *TCP2A* was allocated a large maximum window size to improve its throughput. This window size is apparently two large to fit in the buffers left unoccupied by *TCP1A*'s packets, and *TCP2A* simply oscillates between congestion recovery and performance optimization.

Again, Random Drop can prove useful in breaking *TCP1A*'s grip on the scarce resources and enforcing a more fair policy toward the equal-demand users. A fair treatment would require that both connections be penalized for the bottleneck's congestion and thus both be considered when dropping packets to relieve it. This behavior is exemplified in Figure 4-6 showing the buffer utilization for *PPLINK1* in the *GW1 - GW2* direction when Random Drop is employed. The improved fairness of Random Drop results in a tripling of *TCP2A*'s throughput at the expense of 15% lower throughput for *TCP1A*. It should be mentioned here that even with this improved fairness, *TCP1A* will still attain much higher throughput than *TCP2A*. This is largely due to the fact that the two connections have different end-to-end delay. The effect of this factor will be investigated below as one of the shortcomings of Random Drop.

4.2.1.3 Analysis

The common phenomenon witnessed in the previous experiments is sometimes referred to as the segregation effect. In [DKS89], it is described as a segregation of the sources into "winners, who consume a large amount of bandwidth, and losers, who consume very little." The circumstances in which the segregation occurs differ from one case to another, but they all describe scenarios in which few flows are allowed enough time to stabilize before others join in and share the bottleneck resources. While stabilizing, they build up a certain buffer utilization and monopolize it for the lengths of their durations. The reason why the buffer utilization is constant is that every packet that arrives at a bottleneck resource is simply replacing one that just left, and thus, will find an available buffer.

The deadlock described above occurs because of a lack of randomness in the network, a lack of a mechanism to break the chain of events that yields the undesirable behavior. Random

Drop introduces this much needed mechanism. By choosing a random packet to be dropped upon buffer overflow, it will become possible to break the grip of the few connections tying up the bottleneck's buffers, giving other unlucky connections the chance to utilize them.

It is no question that Random Drop should lead to better performance than No Gateway Policy. How much better, though, is a question that requires testing Random Drop in real networks and observing how often do segregation effects arise. One of the important situations in which it could arise occurs in network environments running Telnet and FTP services concurrently. [DKS89] tested such a scenario and reported that the combination of a 4.3BSD$^+$ flow control with No Gateway Policy "produces fair bandwidth allocation among the FTP sources, but the Telnet sources are almost completely shut out." The frequency of the interaction of FTP and Telnet users alone might make it worthwhile to implement Random Drop.

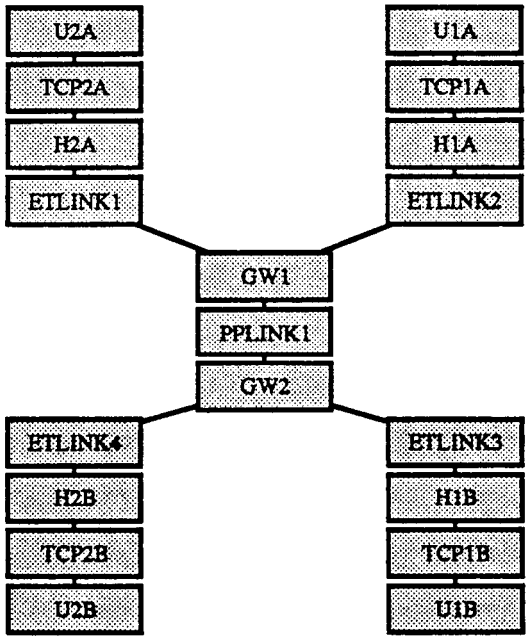


Figure 4-1: Topology III

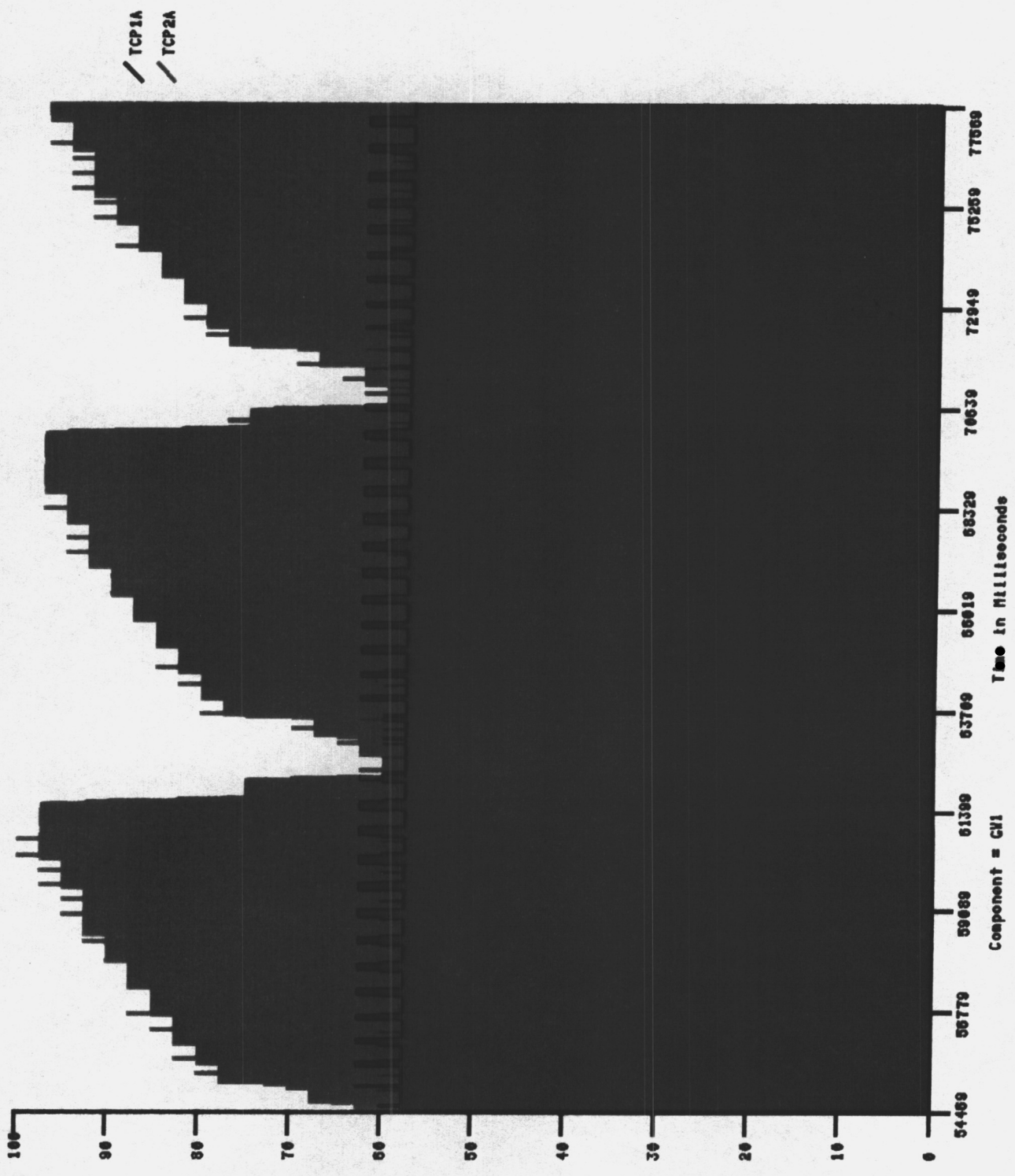


Figure 4-2: Buffer distribution for *PPLINK1* in the *GW1 - GW2* direction using No Gateway Policy.

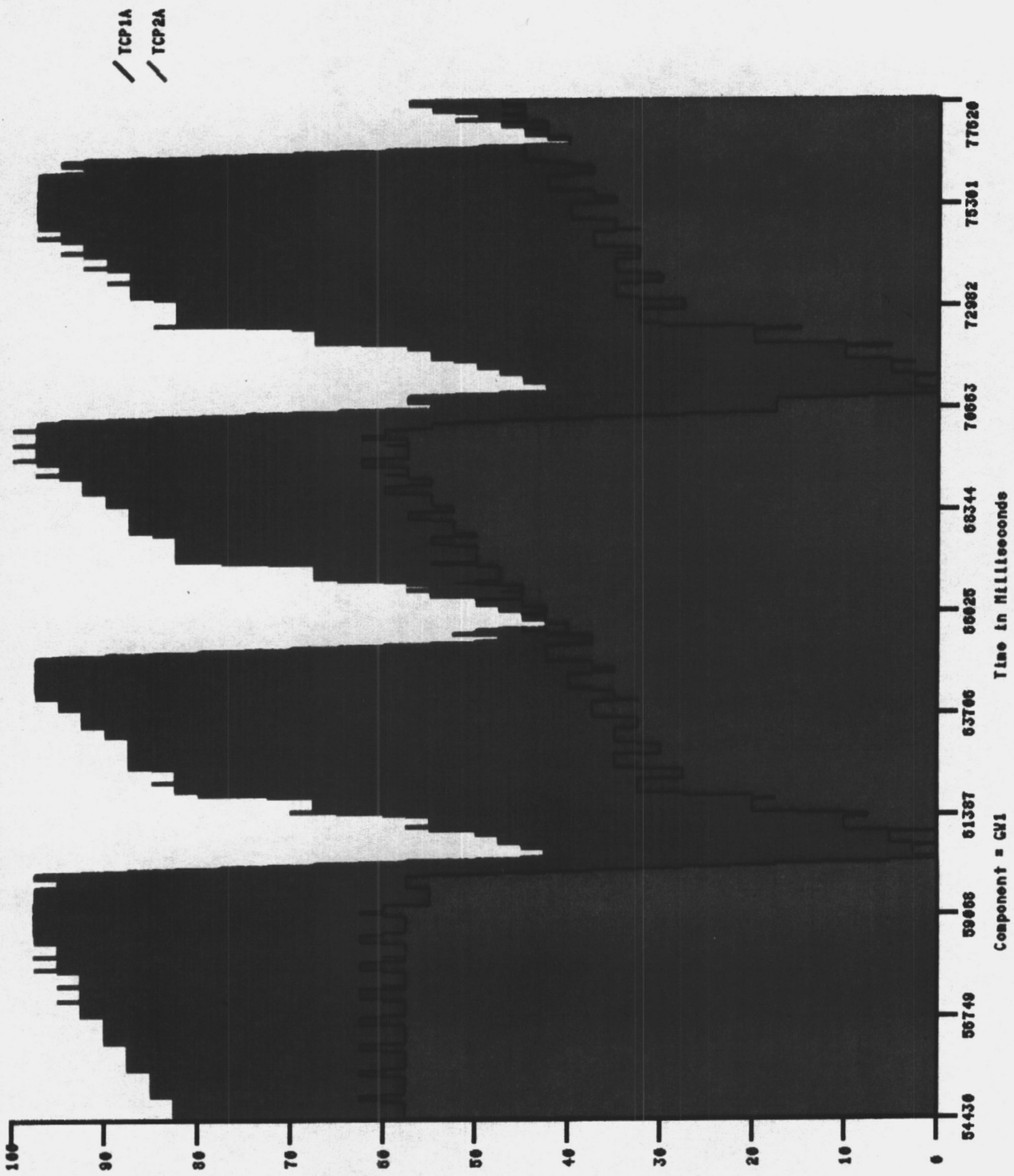


Figure 4-3: Buffer distribution for *PPLINK1* in the *GW1 - GW2* direction using Random Drop.

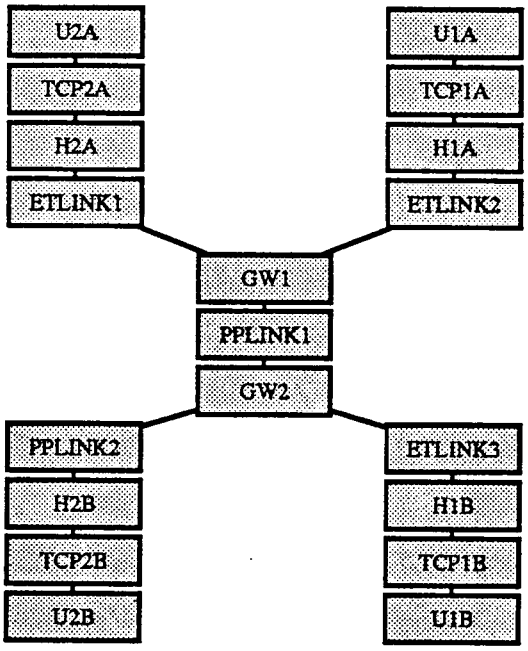


Figure 4-4: Topology IV

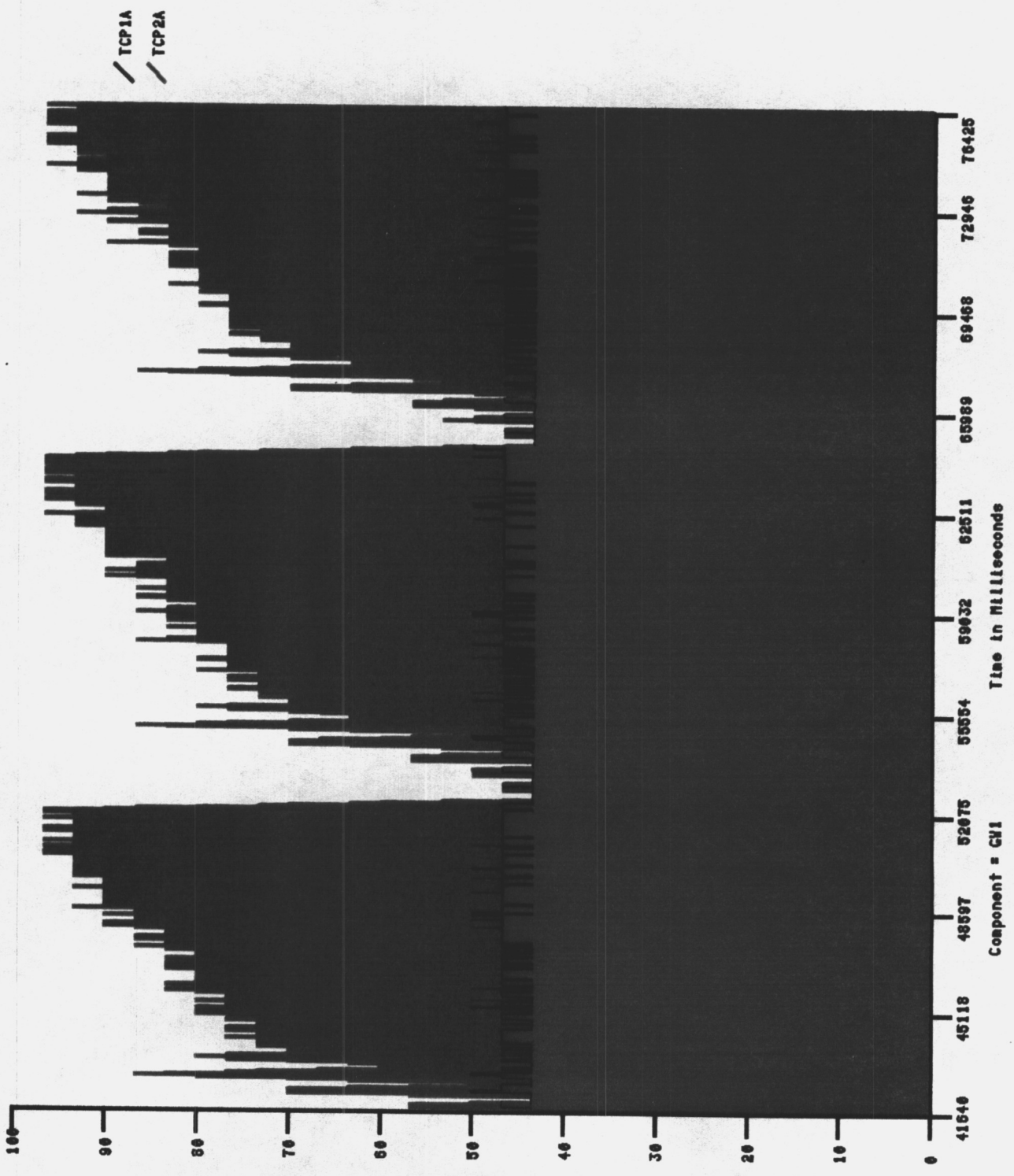


Figure 4-5: Buffer distribution for *PPLINK1* in the *GW1 - GW2* direction using No Gateway Policy.

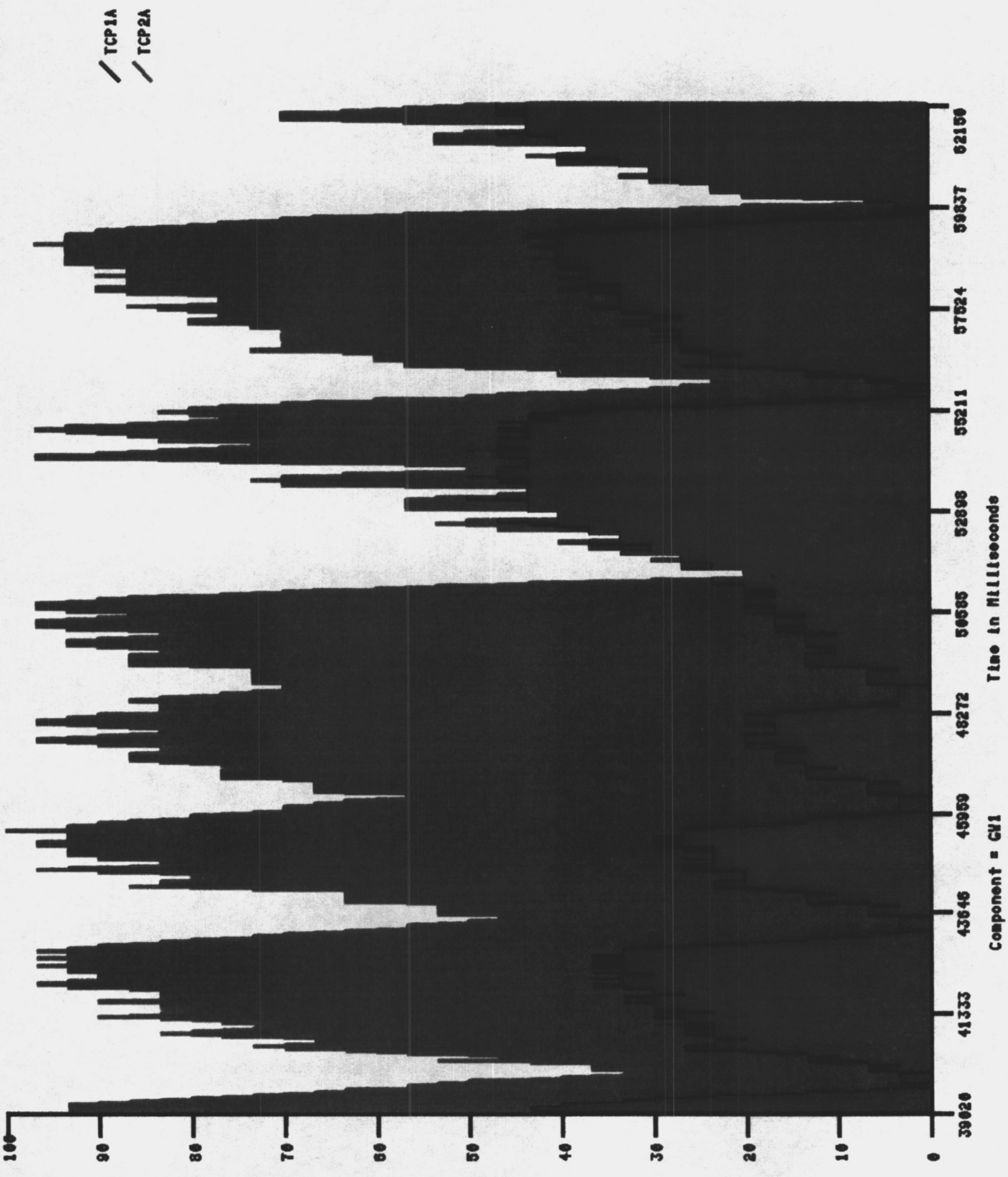


Figure 4-6: Buffer distribution for *PPLINK1* in the *GW1* - *GW2* direction using Random Drop.

4.2.2 Shortcomings of Random Drop

The experiments included here aim at testing Random Drop in typical network environments rather than in specific scenarios, as those described in the previous section. They demonstrate the behavior of Random Drop in a variety of environments supporting heterogeneous connections. The heterogeneity differs from one experiment to another, as indicated by the experiments' names, with each modeling a different aspect of a real TCP connection. The role of the TCP in determining the fair allocation of network capacity will become clear as the experiments are analyzed.

In contrast with the connection model of the previous experiments, the TCP connections here are allocated maximum window sizes large enough to insure that the window never reaches this static boundary. Instead, the TCP congestion and avoidance mechanisms adjust the window size dynamically in response to network feedback about its congestion level. This insures that the TCPs will take full advantage of SLOW START and the congestion avoidance measures to optimize their performance.

4.2.2.1 Experiment 1: Unequal End-to-End Delays

The Internet is a collection of multi-scale networks ranging from the inter-office local area to the cross-continental long-haul. Thus, the end-to-end delay of a connection could be a few milliseconds or several seconds, depending on the propagation, transmission, and queueing delays across its path. Topology II shown in Figure 2-2 models such a realistic network. It supports 14 connections that are identical in every respect except for their end-to-end delays. The point-to-point links were assigned equal propagation and transmission delays to simplify the comparison of the connections' path lengths by mostly counting the number of hops from source to destination[3].

The topology of Figure 2-2 has several bottleneck links due to the intersection of traffic at several gateways. By inspecting the paths of the connections, three main bottlenecks may be identified: *pplink2*, *pplink3*, and *pplink4*. Figure 4-10 shows the bandwidth distribution for

[3]The queueing delays vary from one link to another depending on the traffic load.

pplink3 in the *SAC-GW - UWISC-GW* direction. Even though *pplink3* services the users of 6 TCP connections, more than 60% of its capacity is utilized by *tcp-2a*, which incidentally has the shortest end-to-end delay of the six connections. In comparison to the 1-hop *tcp-2a*, the other connections' path lengths range from 4 to 7 hops. The bandwidth utilization shown in the figure is not exactly proportional to the hop count of each connection, because the queueing delays differ from one hop to the other. For example, the 4-hop *tcp-7a* achieves noticeably higher bandwidth utilization than the 5-hop *tcp-5a*, mostly because it does not cross the bottleneck *pplink4*, avoiding its high queueing delays.

Apparently, the end-to-end delay of a connection greatly influences its utilization of the bottleneck capacity. An important question to ask here is whether *tcp-2a* does actually achieve better throughput, or it merely spends its time retransmitting the many packets dropped by the bottleneck in at attempt to signal it to slow down. Actually, the simulation results report comparable retransmission percentages across all connections. Moreover, *tcp-2a* manages to achieve throughput that is 5-8 times that of the other connections. In summary, *tcp-2a* does achieve better overall performance than the other connections utilizing *pplink3*, even though all connections service users with equal demands. Thus, Random Drop has failed to distribute the utilization fairly among connections with different end-to-end delay, giving preference to the shorter connection and allowing it to grab a sizable portion of the bottleneck capacity.

4.2.2.2 Experiment 2: Mixed Packet Sizes

Real networks support users of varying traffic characteristics requiring different qualities of service. One characteristic that varies from one user to another is the transmission size. For example, Telnet users transmit a few bytes at a time and can fit most transmissions in small packets, while FTP users transfer whole files that need to be packaged in large packets. The effect of mixed packet sizes on the fairness of Random Drop will be tested by changing the packet size for one of the connections in Topology I. This variation does not attempt to model any particular type of user, but rather to isolate the effect of packet size from other factors that might influence the fairness of Random Drop.

Tcp-9a in Topology I uses larger packets than the other connections, specifically 5 times larger. Again this heterogeneousness among the connections should not make a difference as far as the fair distribution of bottleneck capacity among them. An examination of Figure 4-20, though, reveals a different result. The bandwidth distribution of the *US-pplink* in the *CA-GW - MA-GW* direction shows an obvious domination by *tcp-9a*. The high utilization achieved by *tcp-9a* enables it to attain a throughput that is 3-5 times higher than that of any other connection at no extra retransmission cost.

The environments used to test Random Drop here and in the previous section demonstrate variables that are very characteristic of real networks, namely unequal end-to-end delay and mixed packet sizes. So far, Random Drop has failed to provide fair allocation even in environments supporting well-behaved 4.3BSD$^+$ TCP connections. This implies some fundamental problems with the congestion control policy, perhaps ones relating to its underlying assumptions.

4.2.2.3 Experiment 3: Aggressive TCPs

Even though the 4.3BSD$^+$ TCP has improved greatly on the performance of previous implementations, it is not the only one employed in the Internet. Some of the older Berkley distributions as well as other vendor implementations continue to be used. Fair allocation of capacity among users employing different TCP implementations is one the proposed potentials of Random Drop.

The concern for fair allocation in the presence of heterogeneous TCPs stems from the tendency of some vendors to tailor their TCP implementations to provide the user with better service, disregarding the needs of others. One possible implementation will be used here to test the effectiveness of Random Drop in the face of such aggressive TCPs. It will be called the Greedy TCP to distinguish it from the well-behaved 4.3BSD$^+$ TCP.

The Greedy TCP combines some of the algorithms employed by other TCP implementations to achieve better performance. It uses the GO-BACK-N retransmission strategy of the original 4.2BSD TCP and the retransmit timeout calculation of the 4.3BSD$^+$ TCP. Through the frequent retransmissions generated by theses algorithms, the Greedy TCP attempts to squeeze as much

data as possible through the bottleneck, in the process exerting a very heavy load on the network.

Tcp-10a in Topology I was modified to run a Greedy TCP implementation, while the other connections remain to be 4.3BSD$^+$ TCPs. Figure 4-23 illustrates the bandwidth distribution for the *US-pplink* in the *MA-GW - CA-GW* direction and Table 4.1 shows the throughputs and retransmission percentages for all ten connections sharing the link. Despite the equal demands of the five connections sharing the *US-pplink* in that direction, the Greedy *tcp-10a* captures about 50% of the bandwidth. Moreover, even though 30% of its bandwidth utilization is wasted on retransmissions, the Greedy TCP achieves twice the throughput its user is entitled to. Of course in the process, it hurts the other connections greatly. Besides increasing their retransmission percentages, it decreases their bandwidth utilizations to half of their fair shares.

The Greedy TCP might sound too aggressive and misbehaving. Alternatively, consider modifying the 4.3BSD$^+$ TCP to "use a higher gain" when doing SLOW START congestion avoidance [Man89]. The resulting TCP would not generate as many retransmissions as the Greedy TCP, but it will open its window faster than the lower-gain TCPs and thus have a head start on the others. Actually this effect would be very similar to that seen in the previous experiments. This similarity is not accidental but is rather an indication some common shortcomings of Random Drop, as will be explained shortly.

4.2.2.4 Analysis

The previous experiments have described realistic network environments, in which Random Drop has failed to provide the necessary fair allocation of bottleneck capacity among equal-demand users. This failure defies the common intuition about the inherent fairness of Random Drop and prompts a review of the reasons that led to it. It is hoped that they can shed some light as to where the assumptions on which Random Drop is based have gone wrong. The section concludes by a comparison between the behavior of Random Drop and No Gateway Policy in the above network environments to observe any possible merits that the former might have over the later.

Why Random Drop has Failed?

From the experimental results, three major causes of the failure may be outlined.

1. Random Drop chooses packets to be dropped by inspecting the buffer distribution only at the time of overflow, disregarding all previous history. Unfortunately, this does not prove to be a good decision. A comparison between the average buffer distribution upon overflow and the average bandwidth distribution over all times (Figures 4-11 and 4-12) show an obvious disparity between them[4]. Thus, the resource distribution that Random Drop sees is not the actual one achieved.

   By nature, TCP/IP networks are unstable and are subject to various packet clustering phenomena, as described in Chapter 3. Over time, the buffer utilization at a bottleneck resource is likely to oscillate heavily between congestion recovery and performance optimization (Figure 4-13). The periods between buffer overflow events are usually characterized by different flow patterns than those represented by the buffer contents upon overflow. For example, connections with short round-trip delays (i.e.*tcp-2a* in Topology II) recover faster from congestion than connections with longer delay (i.e.*tcp-14a*), Figures 4-7 and 4-8. The recovery period, during which such phenomenon manifests itself, occurs following buffer overflow during intervals characterized by low buffer utilization (Figure 4-9). Thus, disregarding the buffer distribution between overflow events will lead

[4]The disparity between the two graphs is due to two combined effects: different end-to-end delays; and different packet sizes (acknowledgements have zero length). The two will be isolated below in the discussion of the assumptions of Random Drop.

Random Drop to make inaccurate estimates of the average buffer distribution over all times.

2. Since packets of different sizes use equal-size buffers, they will see equal drop probabilities, even though, they utilize bandwidth differently. In the Mixed Packet Sizes experiment, *tcp-9a*, which uses much larger packets than the other connections, attains a very average buffer utilization while using up more than 40% of the total bandwidth (Figures 4-18 and 4-19). Thus, Random Drop will unfairly favor the connections with larger packets, subjecting them to fewer packet losses upon congestion.

3. When relieving congestion, Random Drop causes different connections to lose different numbers of packets depending on their buffer utilizations upon congestion. Nonetheless, they all have to recover, and the recovery effort is fairly similar, almost regardless of how much was lost. This is due to the fact that the TCP congestion recovery overhead is far higher than the average retransmission cost of a packet. The overhead consists of the timeout period plus the time spent SLOW STARTing following a packet loss, which is usually several round-trips of substantial propagation delay. Thus, the fact that a connection lost something is much more important than how much was actually lost.

   The connections in Topology IV compete for the capacity of the bottleneck *PPLINK1*[5]. Despite their equal demands, they exert different loads on the network depending on their end-to-end delays. With Random Drop gateway congestion control, *TCP1A* loses 25 packets in 13 buffer overflow events; whereas, *TCP2A*, with the much higher end-to-end delay, loses 8 packets within 8 buffer overflow periods. Even though the faster connection is being penalized more heavily (losing more packets), it is losing them in fewer congestion (buffer overflow) periods, and thus SLOW STARTing less often. One way to think of this is that the higher is the *number of SLOW STARTs / number packets lost* ratio for a connection, the more often it gets hit for the same congestion contribution.

What is Wrong with the Assumptions?

There are two main assumptions underlying Random Drop: the buffer distribution of a

[5]Both connections are assumed to be using very large maximum window sizes.

resource is proportional to its bandwidth distribution; and the drop probability is proportional to the average user flow rate.

Whether the buffer distribution is actually representative of the bandwidth distribution or not is not the question, since Random Drop is based on the buffer distribution upon overflow not over all times. In the previous section, it was shown that the buffer distribution upon overflow does not necessarily agree with the average bandwidth distribution achieved. One reason for this is that data and acknowledgement packets occupy equal-size buffers and get treated similarly by Random Drop, even though acknowledgements consume negligible bandwidth compared to data.

Another reason follows from the difference in congestion recovery time between different connections depending on their end-to-end delay. Recall that shorter connections recover faster, getting a head start over other connections which enables them to squeeze more data through over time. Topology II was run without any of the connections passing through *pplink3* in the *UWISC-GW - SAC-GW* direction being active, in order to eliminate any acknowledgements flowing in the opposite direction and consuming part of the available buffers. Figures 4-15 and 4-14 illustrate the average buffer distribution upon overflow and the average bandwidth distribution for *pplink3*. The discrepancy between the two distributions reflects the bandwidth gain achieved by a connection during periods of no congestion, and thus concealed from Random Drop. For example, the 1-hop *TCP-2A* achieves 11 times the bandwidth utilization of the 5-hop *TCP-5A*, whereas it consumes only 8 times its buffer capacity during overflow periods when Random Drop is activated.

Even if we did assume that the buffer distribution upon overflow accurately represents the average bandwidth distribution, the Random Drop probability does not seem to be proportional to the average user flow rate. For one thing, the TCP flow control alters the effective user load on the network. The TCP flow pattern is a product of the user data arrival function, the TCP maximum window size, and the TCP flow control and avoidance schemes. To simplify the analysis, the users are equal-demand file transfers, and the connections use equal maximum window sizes, large enough so that the flow is controlled only by the TCP dynamic window adjustment. Thus, the TCP congestion control and avoidance mechanisms solely determine the

user flow rate into the network according to the following formula

$$\textit{average TCP data transmission rate} = \frac{\textit{average window size} \times \textit{average retransmission rate}}{\textit{average round trip delay}}$$

The simulation experiments, conducted above, illustrate the effect of varying any of these three parameters on the user flow rate and the associated bandwidth utilization achieved.

In the first experiment, the various end-to-end delays lead to different average window sizes, due to the dynamic adjustment of the window over varying round-trip times. This results in shorter connections having larger average window sizes and correspondingly faster transmission rates (Figures 4-7 and 4-8). Another way of increasing the average window size is illustrated by the Mixed Packet Sizes experiment. Since the exponential gain of SLOW START is the same across all connections, using larger size packets implies a larger average window size and a correspondingly higher TCP data flow rate, Figures 4-16 and 4-17.

Up until now, the variables used to control the TCP flow rate affected mostly the transmission rate of actual user data. In contrast, the TCP flow can consist of a large number of retransmissions. Even though retransmissions are not usually desirable, they may be used by an aggressive TCP to break the Random Drop policy, since the more duplicate copies pass through a bottleneck, the higher is the probability of at least one of them getting across. The Greedy TCP uses this approach to maximize its throughput. Figure 4-22 illustrates the TCP flow pattern for the Greedy *tcp-10a*, and the speed of its transmission compared to any other connection such as *tcp-8a*, shown in Figure 4-21. The results of this experiment are alarming, since they demonstrate that "it pays to be bad". By behaving aggressively, *tcp-10a* does manage to increase its throughput appreciably, in the process hurting the more well-behaved 4.3BSD$^+$ TCPs greatly.

The results of the previous experiments have one common denominator, by increasing the data flow rate into the network, more bandwidth can be achieved and correspondingly higher throughput. A fast connection might have more buffered packets than a slow connection. However, since all packets have equal probability of being dropped, more packets belonging to the fast connection will escape the Random Drop. As expressed more concisely by a colleague of

mine [Zha89], if any buffered packet may be dropped with probability p, then two connections with arrival rates r_1 and r_2 at the bottleneck will attain throughputs of $(1-p)r_1$ and $(1-p)r_2$. Thus, the ratio of the achieved throughputs is equivalent to the ratio of arrival speeds, and Random Drop fails to penalize the faster connection.

This shortcoming of Random Drop will be evident in any network environment characterized by flows of various speeds. In the experiments conducted above, the TCP flow control mechanism was responsible for the differences in flow rates. Heterogeneous users with different demands can also produce flow rate variations, although they are limited by the TCP flow control mechanisms. Thus, whether the user or the TCP is the limiting factor, the end's flow rate into the network will decide how much bandwidth it will get[6].

4.2.2.5 Comparison with No Gateway Policy

Besides the proposed merits of Random Drop, its appeal follows from an intuitively better behavior than that of No Gateway Policy. The term 'better' here refers to the fairness of the dropping policy. Due to the packet clustering phenomena seen in network traffic, dropping packets in FIFO fashion may result in unfair penalization of a few connections, that might not even be contributing to congestion. On the other hand, using Random Drop, any such unfairness would be eliminated.

The performance of gateway congestion control using No Gateway Policy is illustrated for the network environments of the above experiments in Figures 4-24, 4-25, and 4-26. In comparison to their Random Drop counterparts, these figures do not demonstrate substantial long-term differences. Instead, networks employing No Gateway Policy congestion control seem to exhibit more dynamic short-term behavior with larger variations in bandwidth utilization. These variations are a result of the sensitivity of the dropping policy to the Local Packet Clustering Effect, described in Chapter 3.

Due to the SLOW START TCP retransmission strategy, the packets of a connection will

[6]This is the reason why it was reasonable to simplify the analysis by assuming equal-demand users. The flow rate variations provided by the TCP were sufficient to demonstrate the failure of Random Drop to provide the required fairness.

exhibit a clustering effect during the exponential window sizing stage of the congestion control policy. Such clusters will increase the immediate load on an encountered bottleneck. Furthermore, If the bottleneck's queues are long enough, the clustered packets will cause them to overflow and the resource to get congested. In such a situation, the connection, to which the packet cluster belongs, will receive most of the penality for the congestion, even if its buffer utilization is within its fair share. Due to the randomness of network traffic, the connection should not continue to be unlucky for long and should resume a stable transmission shortly. This short-term unfairness experienced by the TCP connections manifests itself in the fluctuations in bandwidth utilization seen in the No Gateway Policy bandwidth distribution figures.

Since buffer overflow is fairly common in heavily utilized network environments, the temporary unfairness associated with No Gateway Policy will occur frequently. Nevertheless, it will not be aimed at any particular connection but will rather target different connections in a fairly random fashion. The combined effect will result in long-term performance similar to that of Random Drop. Figure 4-27, for example, demonstrates the average bandwidth distribution achieved by the 5 identical connections sharing *pplink3* in *MA-GW - CA-GW* direction in Topology I. Despite some mild fluctuations in the individual utilizations, there is an apparent identical sharing of the bottleneck link.

In assessing the improvement in fair capacity distribution that could be achieved by Random Drop over No Gateway Policy, a comparison of the characteristic traffic patterns susceptible to drop is in order. Random Drop attempts to identify aggressive users and penalize them by sampling the buffer contents statistically. In doing so, it is assuming that the buffer contents represent the incoming packet traffic fairly well. Even though this was shown to be an inaccurate assumption, the range of packets being considered for drop usually spans most connections. On the other hand, using No Gateway Policy, the distribution of packets overflowing the bottleneck's buffers serves as a sample of the overall incoming traffic stream. The question here is how representative is this sample of the actual user loads imposed on the network.

Besides the Local Packet Clustering effect apparent during SLOW STARTs, the TCP/IP networks do not seem to exhibit any other phenomena that could lead to substantial clustering of packets belonging to one TCP connection. Moreover, as was explained in Chapter 3, the

processing and queueing delays associated with network resources serve to space out clustered packets after each round trip. Since LPC is sustained by SLOW START, the effect will disappear as soon as the dynamic window sizing brings the connection to a stable flow pattern. The dispersion effect of the network resources along with the overlapping of traffic of different connections at cross points injects enough randomness in network traffic to make the drop data sample of No Gateway Policy fairly characteristic. This effect accounts for the similarity in the long-term behavior observed between Random Drop and No Gateway Policy. Thus, the performance analysis of Random Drop, described above, can also be applied to No Gateway Policy congestion control, yielding similar results of failure to provide the necessary fair allocation of bottleneck capacity.

4.2.2.6 Appendix A: Experimentation Figures

The following section contains the experimentation figures obtained through network simulation and used to illustrate the behavior of Random Drop and assess its performance. The figures are grouped according to the experiment they represent, not according to the order in which they are referenced. Since they are referred to repeatedly throughout the text, this represents a simpler and more logical classification.

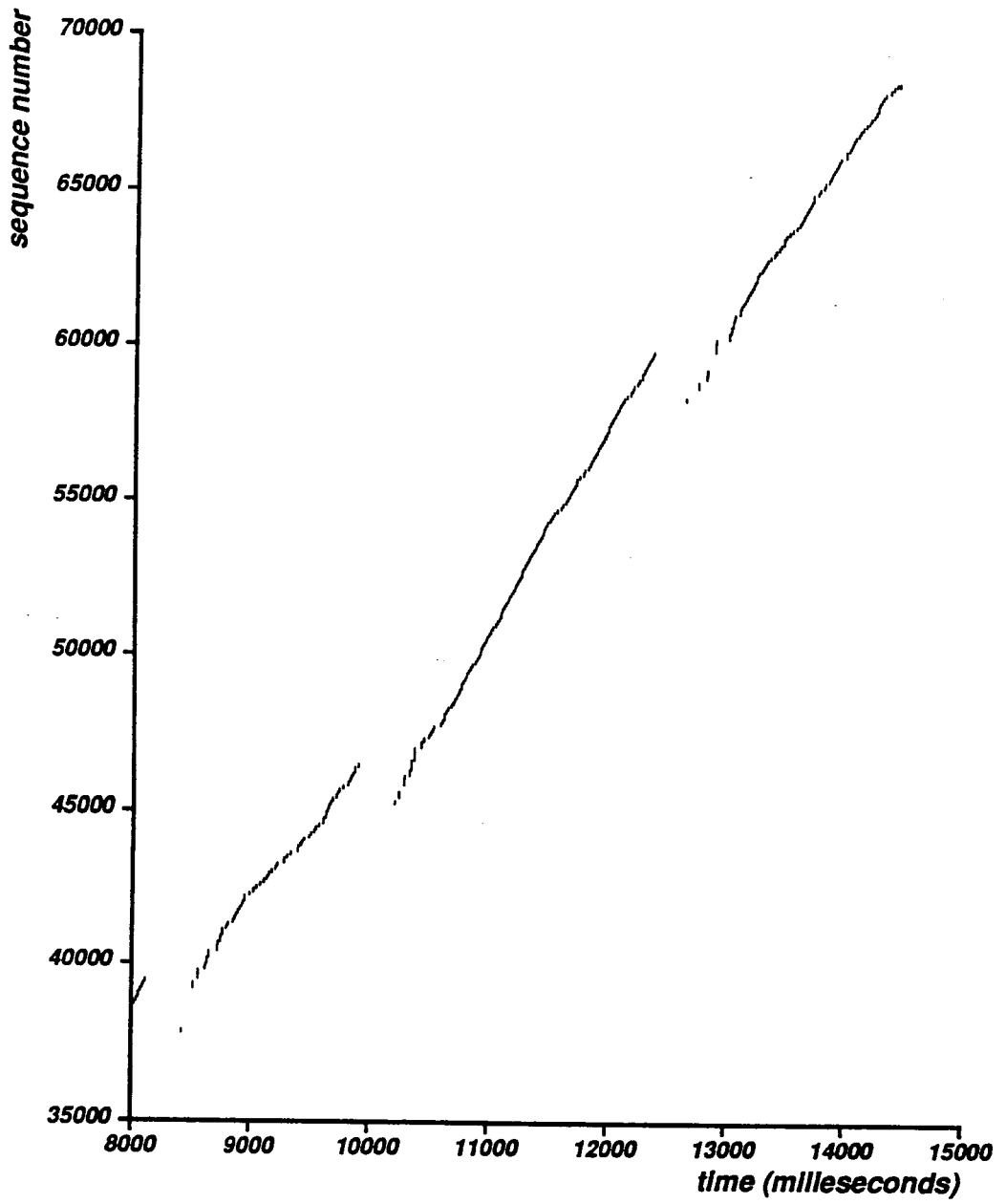


Figure 4-7: Sender sequence number logging for *tcp-2a*.

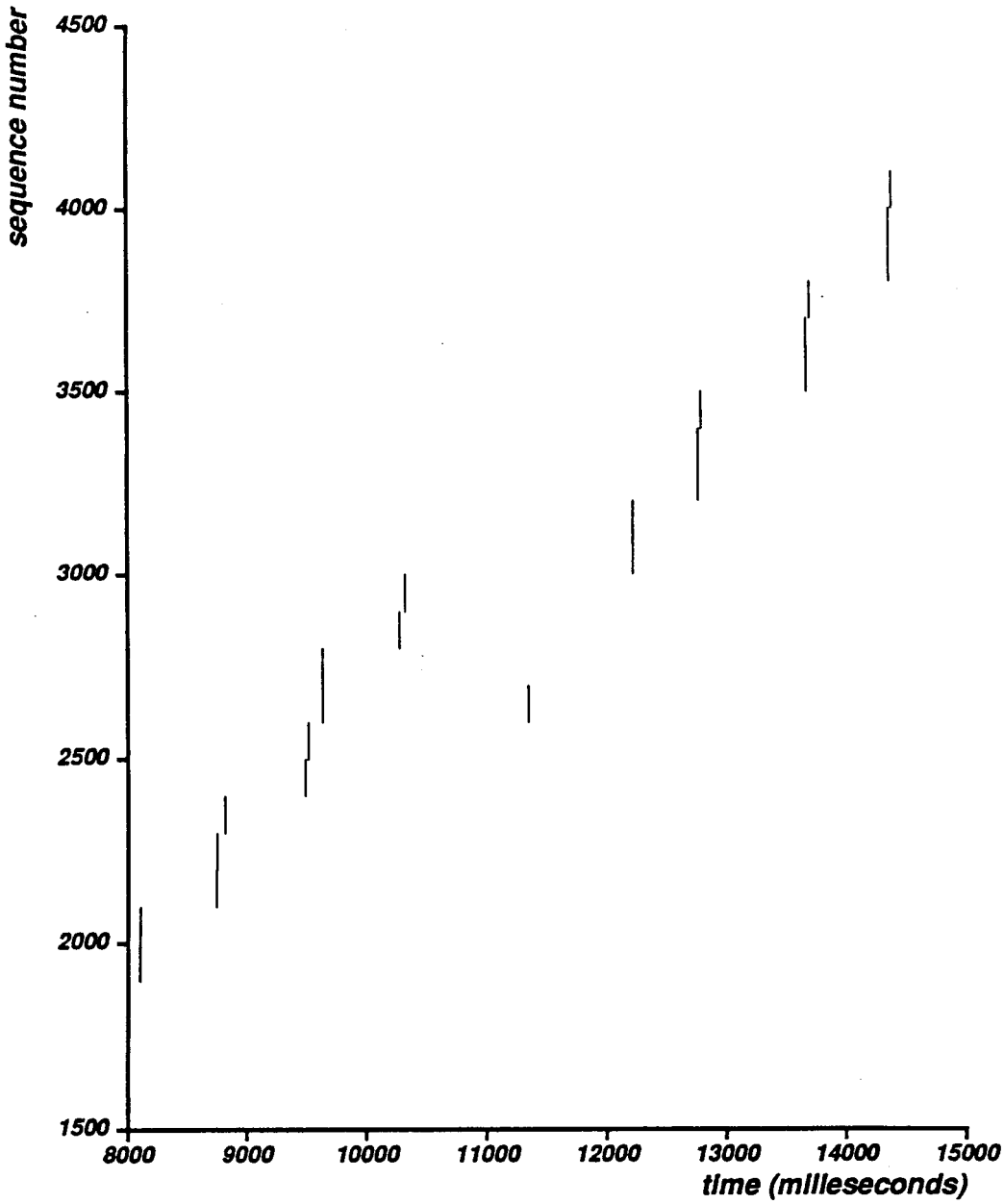


Figure 4-8: Sender sequence number logging for *tcp-14a*.

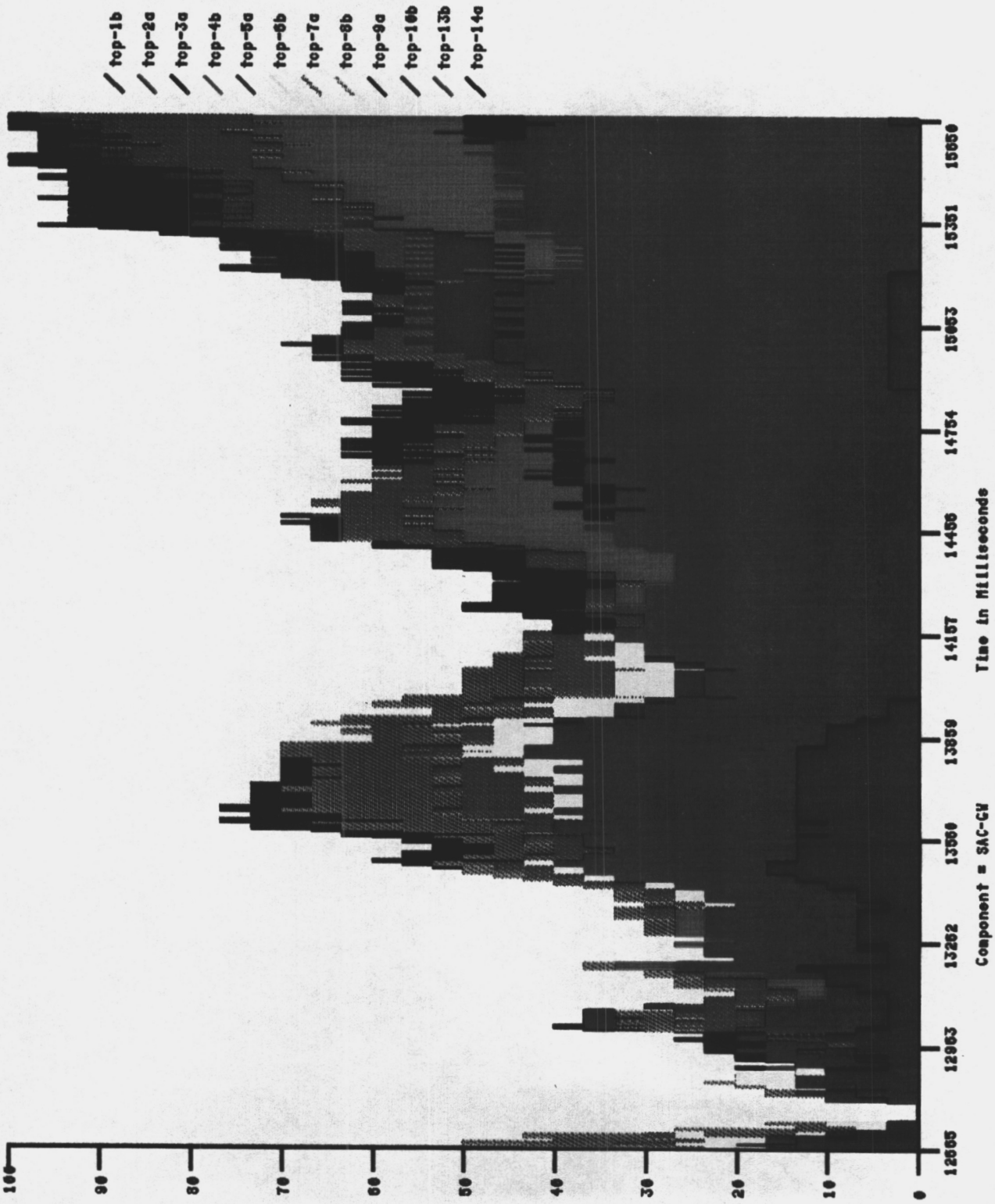


Figure 4-9: Short-term buffer distribution for *pplink3* in the *SAC-GW - UWISC-GW* direction.

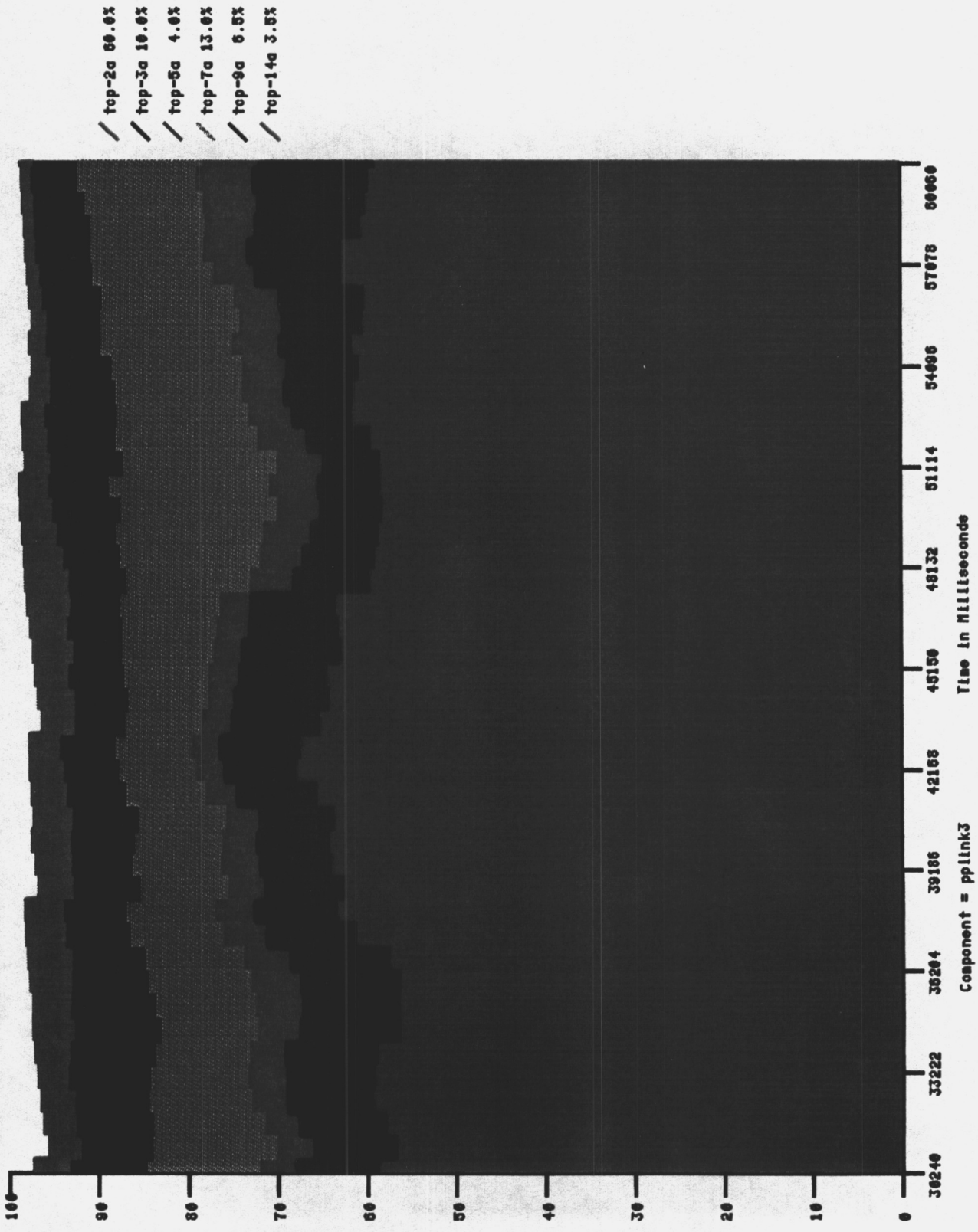


Figure 4-10: Smoothed bandwidth distribution for *pplink3* in the *SAC-GW - UWISC-GW* direction.

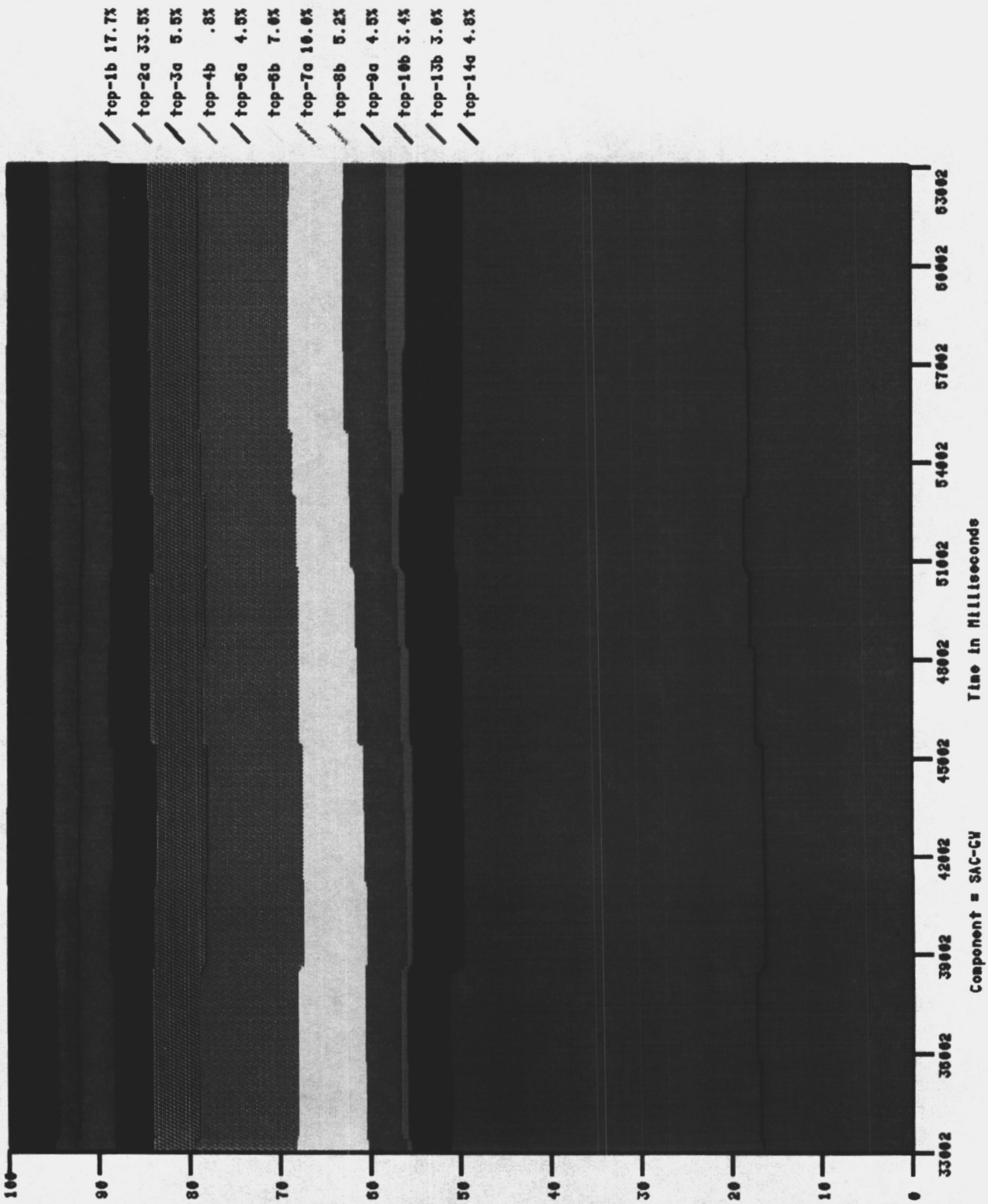


Figure 4-11: Average buffer distribution upon overflow for *pplink3* in the *SAC-GW - UWISC-GW* direction.

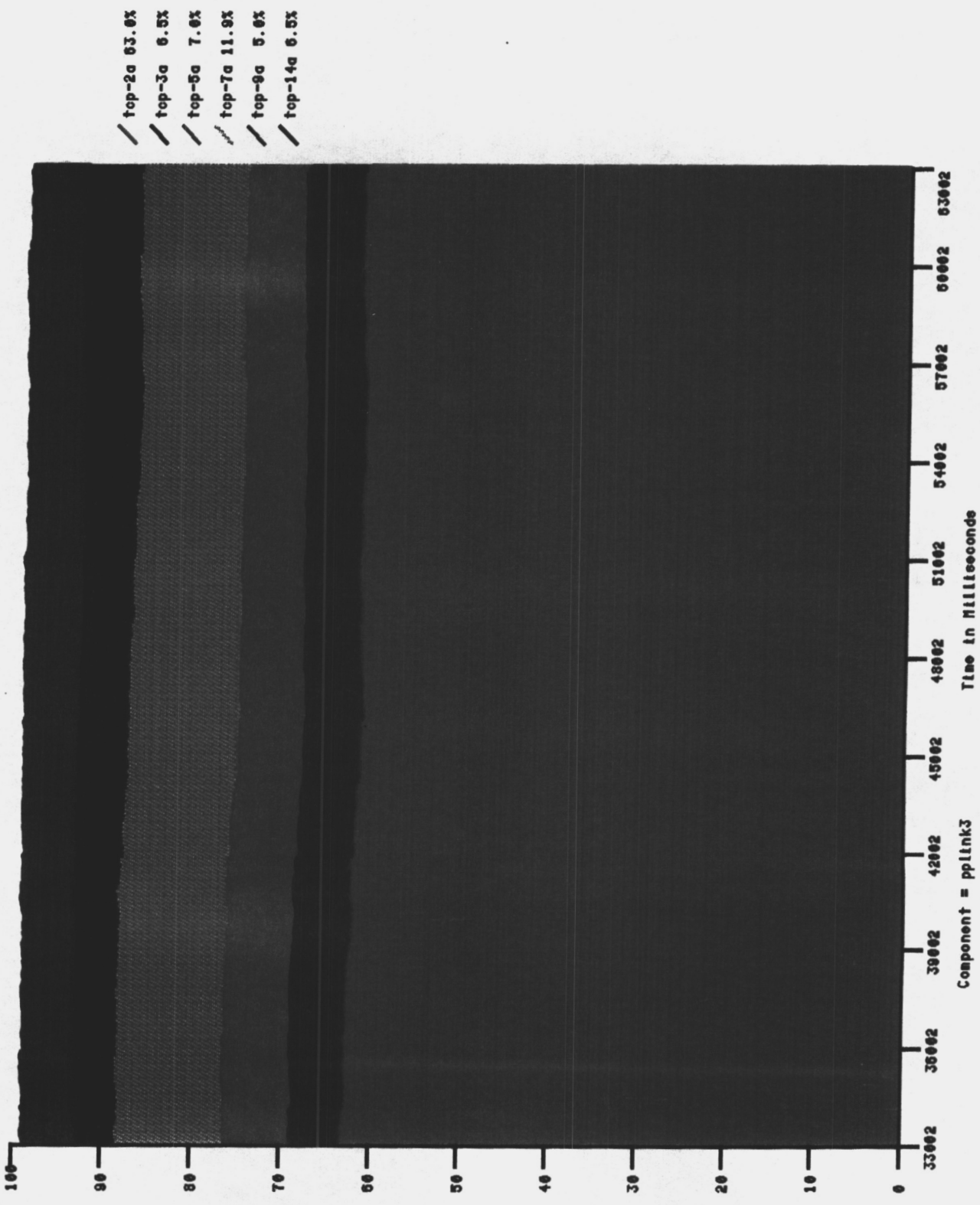


Figure 4-12: Average bandwidth distribution for *pplink3* in the *SAC-GW - UWISC-GW* direction.

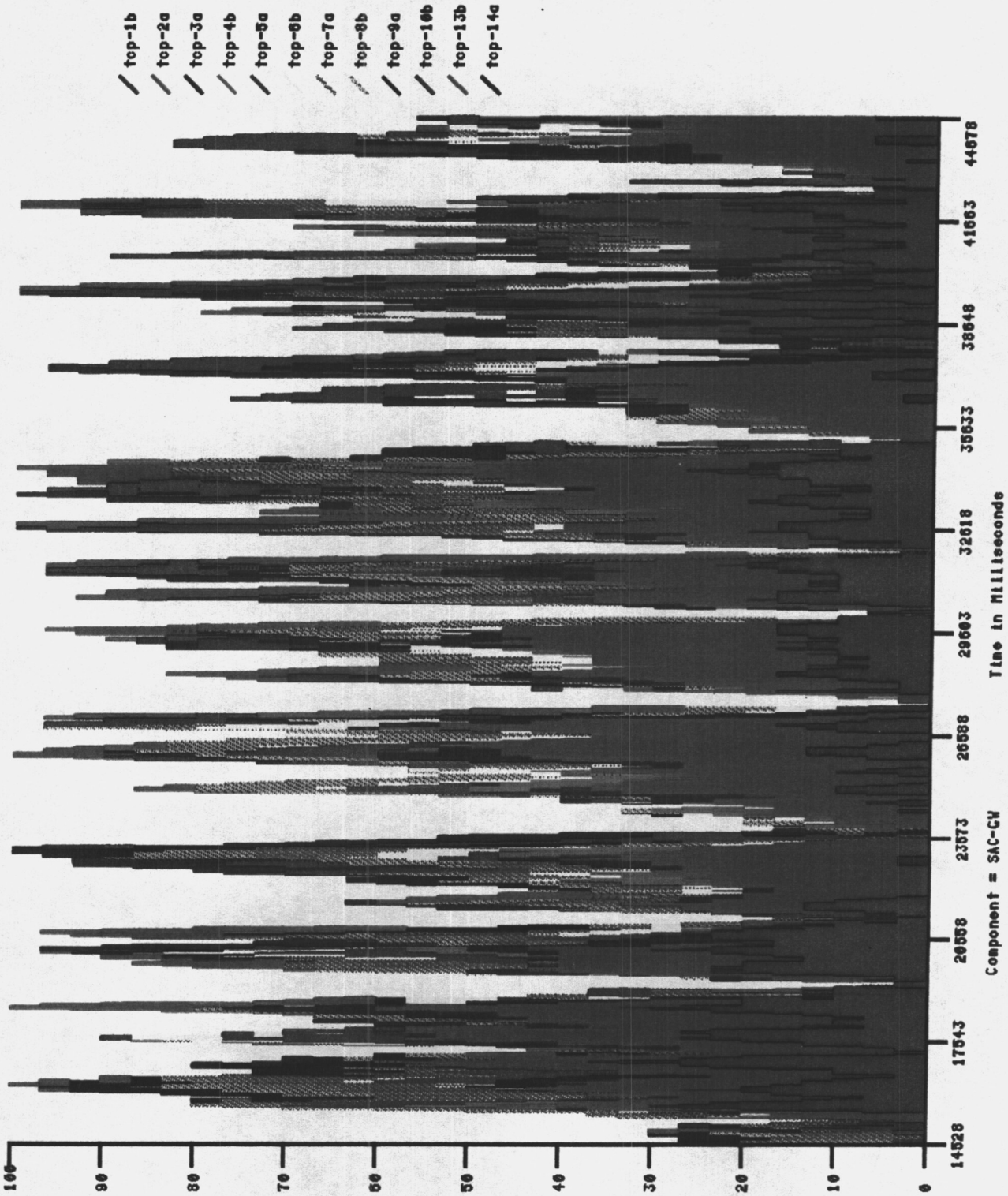


Figure 4-13: Long-term buffer distribution for *pplink3* in the *SAC-GW - UWISC-GW* direction.

Figure 4-14: Average bandwidth distribution for *pplink3* in the *SAC-GW - UWISC-GW* direction.

Figure 4-15: Average buffer distribution upon overflow for *pplink3* in the *SAC-GW - UWISC-GW* direction.

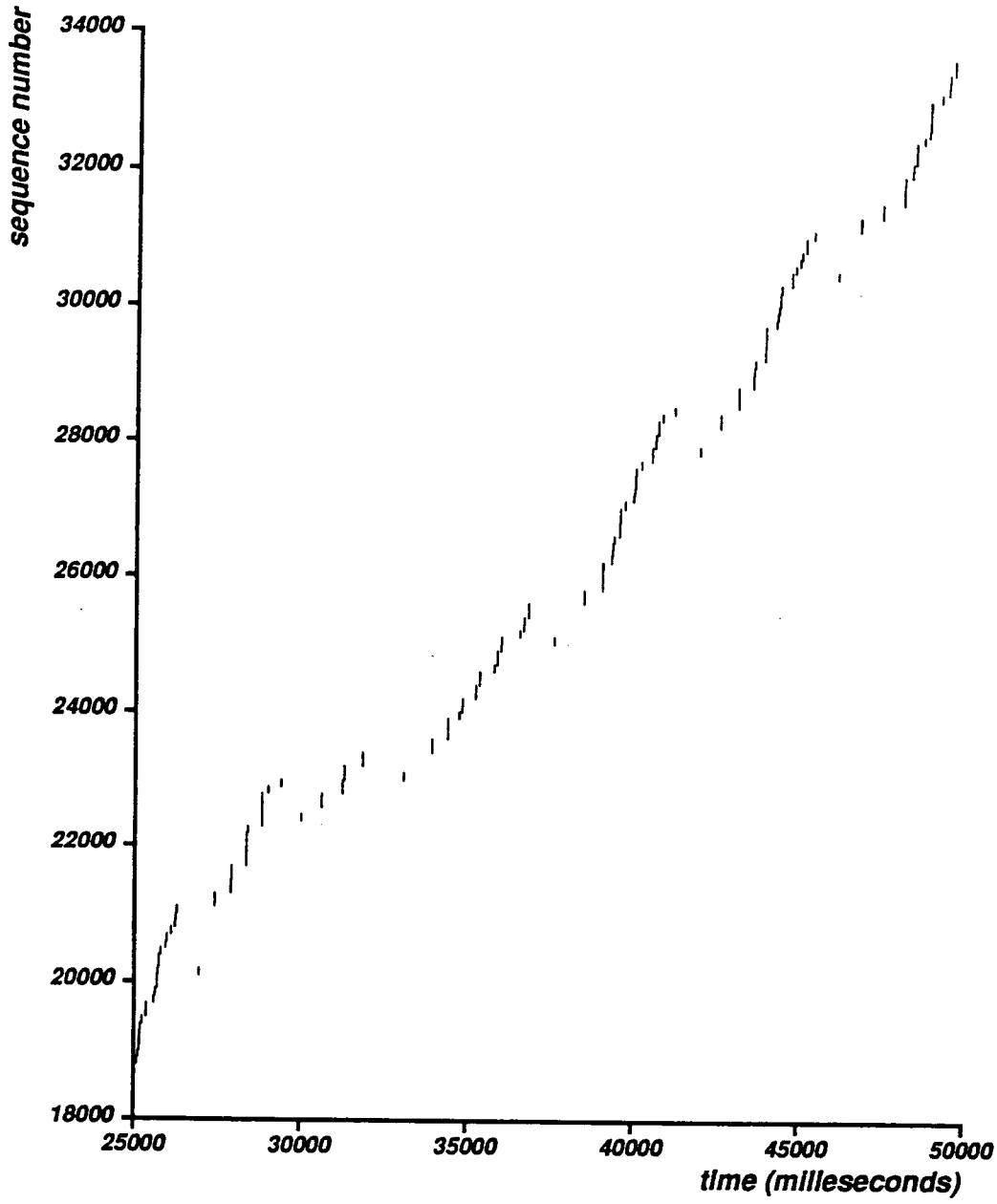


Figure 4-16: Sender sequence number logging for *tcp-7a*.

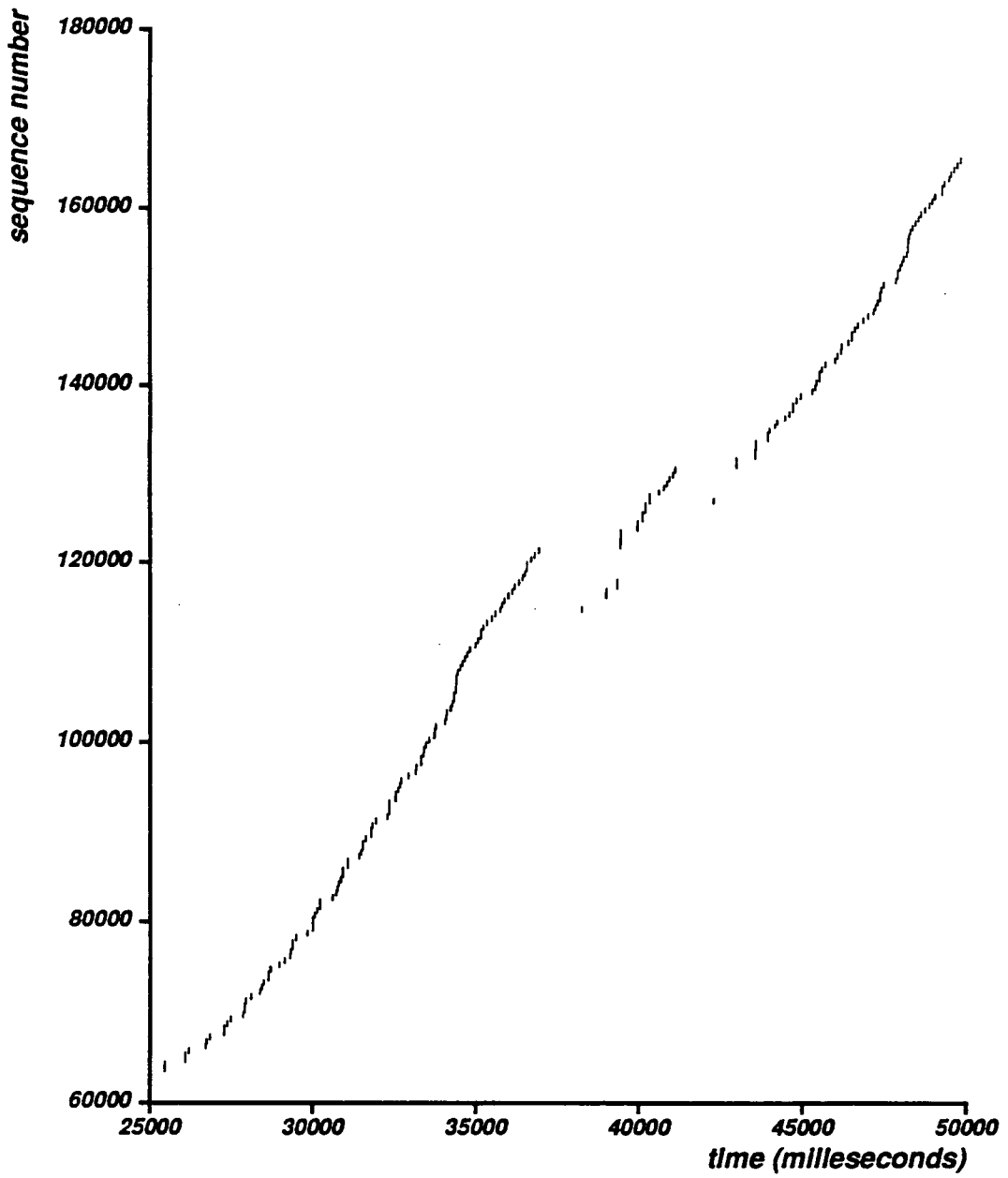


Figure 4-17: Sender sequence number logging for *tcp-9a*.

Figure 4-18: Buffer distribution for *pplink3* in the *CA-GW - MA-GW* direction.

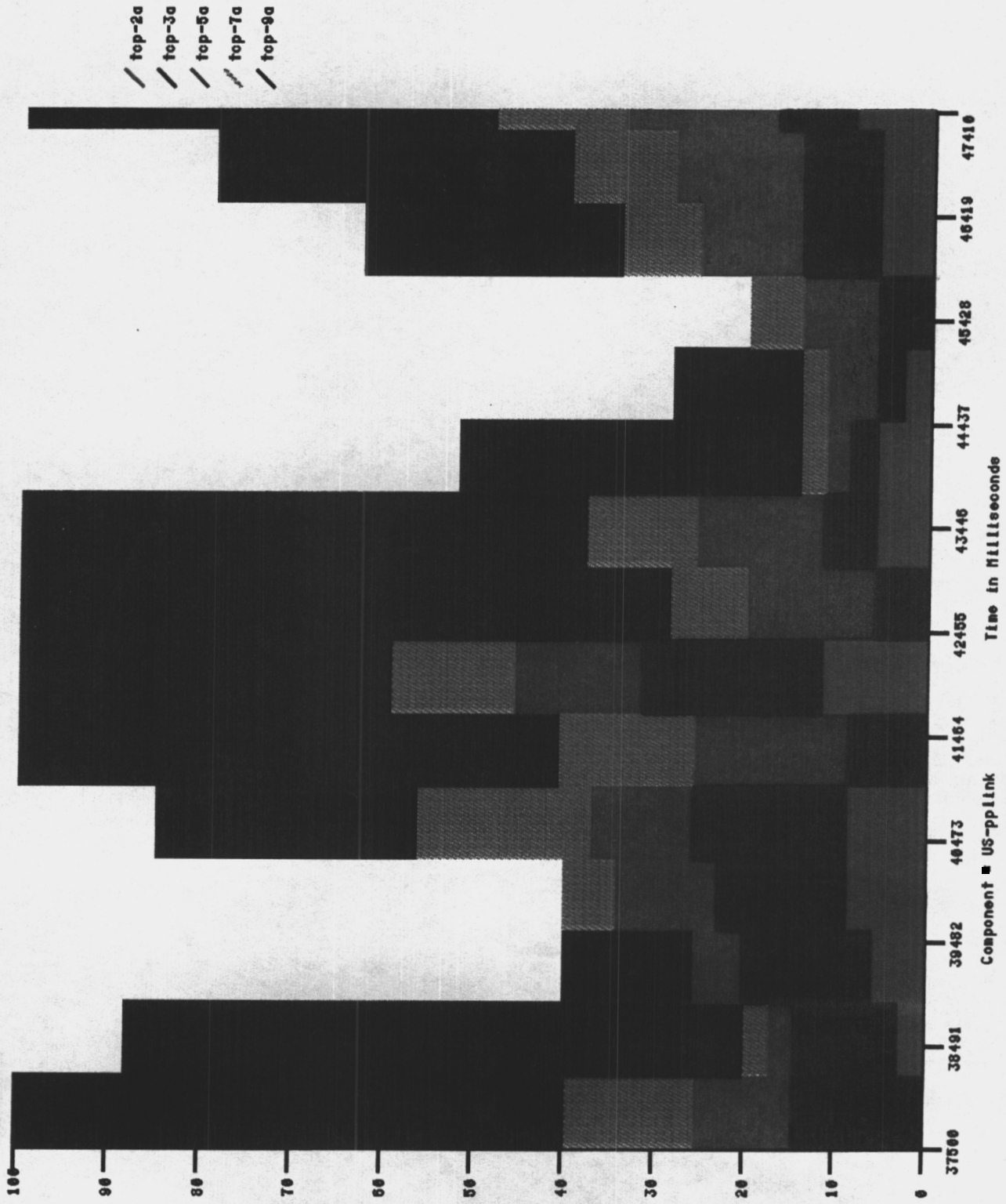


Figure 4-19: Bandwidth distribution for *pplink3* in the *CA-GW - MA-GW* direction.

Figure 4-20: Smoothed bandwidth distribution for *pplink3* in the *CA-GW - MA-GW* direction.

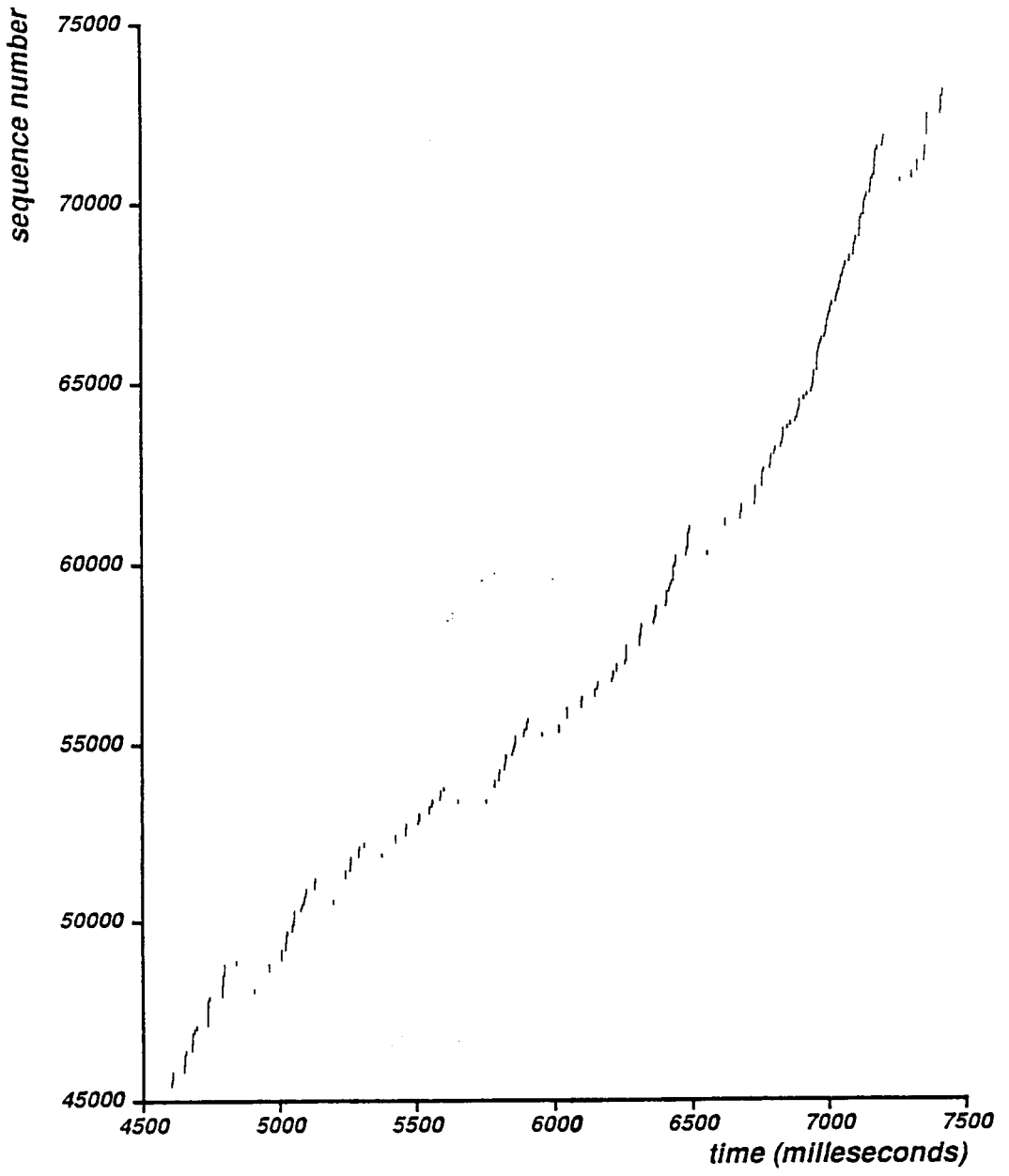


Figure 4-21: Sender sequence number logging for *tcp-8a*.

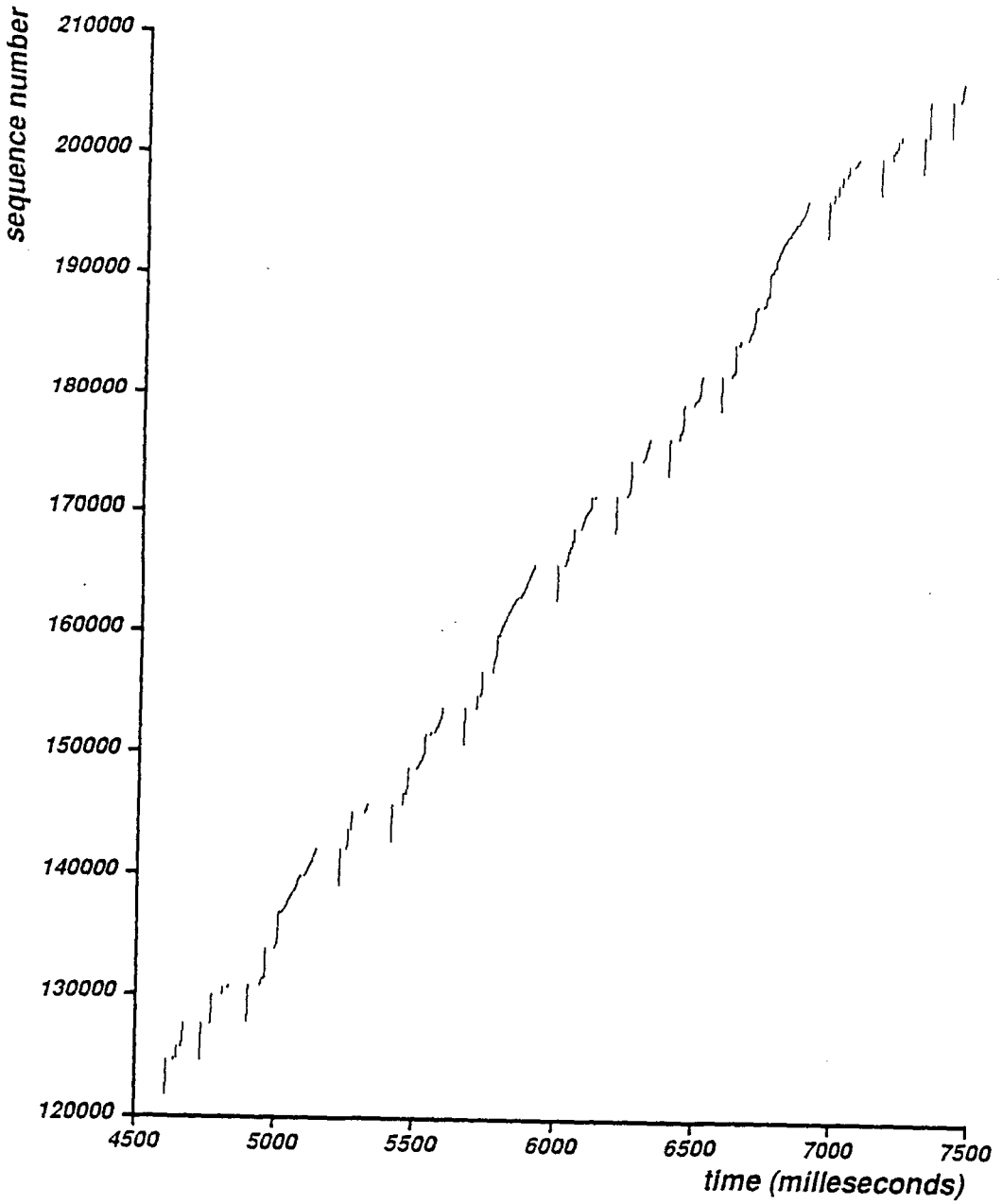


Figure 4-22: Sender sequence number logging for *tcp-10a*.

Figure 4-23: Smoothed bandwidth distribution for *US-pplink* in the *MA-GW - CA-GW* direction.

Connection Name	*Retransmission Percentage*	*Throughput (Kbytes/sec)*
tcp1a	14	429
tcp2a	29	1837
tcp3a	4	1405
tcp4a	11	710
tcp5a	1	1703
tcp6a	19	353
tcp7a	2	1679
tcp8a	7	975
tcp9a	0	541
tcp10a	35	2742

Table 4.1: Throughputs and retransmission percentages of TCP connections.

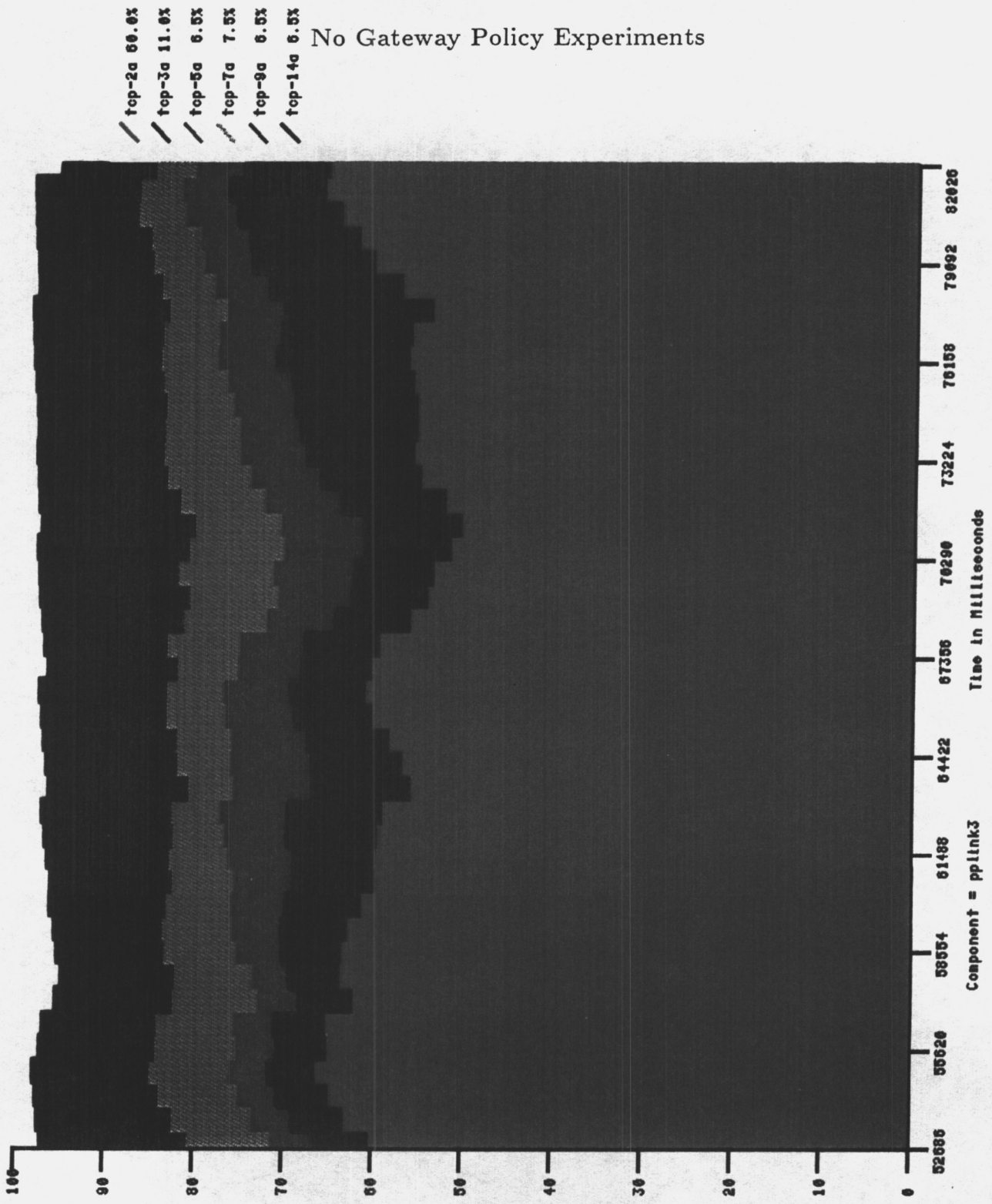


Figure 4-24: Smoothed bandwidth distribution for *pplink3* in the *SAC-GW - UWISC-GW* direction.

Figure 4-25: Smoothed bandwidth distribution for *US-pplink* in the *CA-GW - MA-GW* direction.

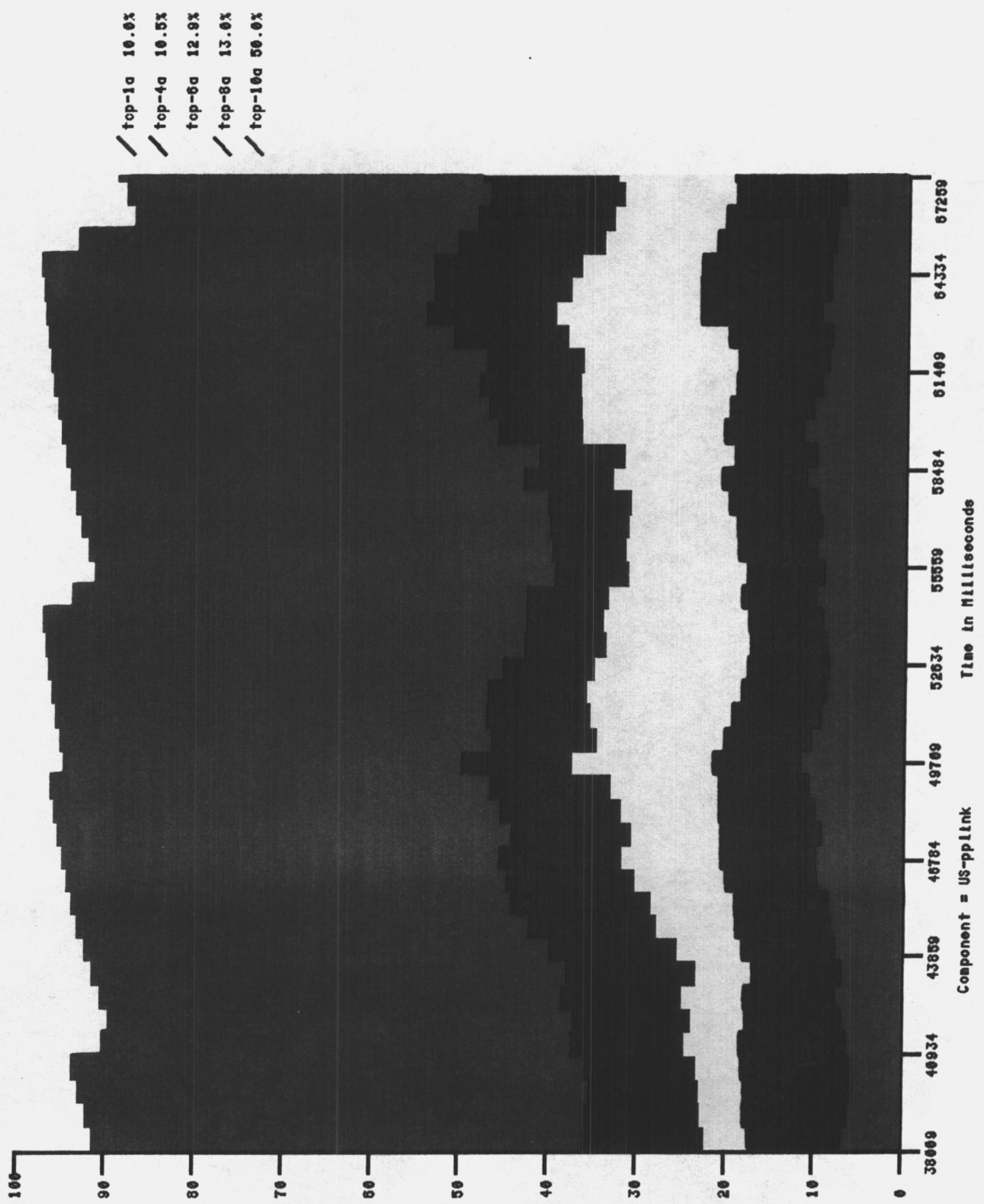


Figure 4-26: Smoothed bandwidth distribution for *US-pplink* in the *MA-GW - CA-GW* direction.

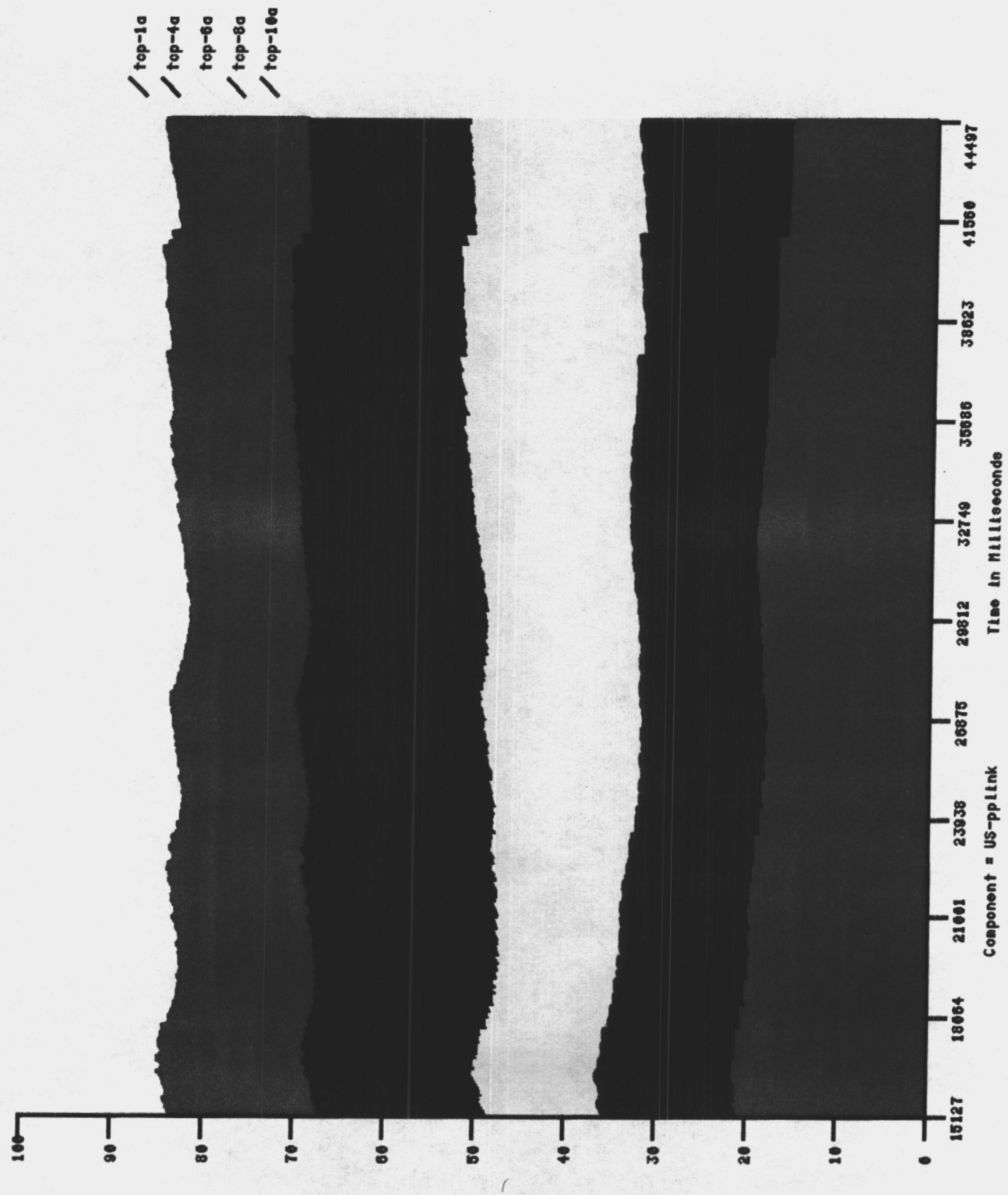


Figure 4-27: Smoothed bandwidth distribution for *US-pplink* in the *MA-GW - CA-GW* direction.

Chapter 5

Early Random Drop

The strong belief in the fairness of Random Drop has inspired a proposal for its modification to perform congestion avoidance functions as well as congestion control. Instead of merely applying the dropping policy upon buffer overflow to relieve and signal congestion, it can also be applied prior to that to indicate its potential occurrence. At some appropriately chosen drop level, Random Drop is applied with a drop probability that reflects the rate of queue growth.

The previous chapter has demonstrated some of the shortcomings of Random Drop, several of which seem to be inherent to the dropping policy itself. In this chapter, Early Random Drop will be tested for its effectiveness as a congestion avoidance mechanism, as well as for its ability to overcome some of the failures of Random Drop.

5.1 Functionality

Early Random Drop is intended to act as a congestion avoidance mechanism that foresees imminent congestion and takes measures to avoid it. The mechanism consists of three functions: predicting that congestion will soon occur, identifying the flows contributing to it, and signaling them to slow down. The last two functions are to be carried out by simple Random Drop applied early upon predicting congestion and before it actually occurs.

Early Random Drop predicts congestion of a resource by monitoring the length of the packet queue awaiting service and observing its growth pattern. Since traffic fluctuations are

fairly normal in a dynamic network environment, those of reasonable magnitude should be safely accommodated and not be falsely mistaken to signal impendent congestion. For this reason, Early Random Drop uses a Drop Level parameter that defines the region of safe traffic fluctuations which are not likely to lead to congestion.

The queue growth pattern provides different signs of approaching congestion, which are then weighed according to their importance. The weight is dictated by the Drop Probability parameter, which reflects how strongly Early Random Drop believes in the congestion indications. Upon exceeding the Drop Level, the queue growth pattern is translated into a congestion prediction and Random Drop is applied to the contents of the queue with the Drop Probability. Expressed more concisely,

IF queue-length > Drop Level
THEN IF get-random() < Drop Probability
THEN drop(packet).

One crude way of predicting congestion is to simply ignore the growth pattern beyond the Drop Level and consider all queue activity from there on to indicate imminent congestion by the same amount and thus be weighed similarly using one uniform Drop Probability. Even though other more refined predictions of congestion exist [1], this one will suffice to analyze some of the important aspects of Early Random Drop and assess its potential performance.

5.2 Objectives

Just as in simple Random Drop, Early Random Drop should conform to the performance criteria of providing fair service among network users while achieving good overall network performance. The previous chapter illustrated how Random Drop has failed to meet these performance criteria on several accounts. There is some reason to believe that Early Random Drop might be able to do better in some situations due to the head start it gets to avoid congestion.

[1] One suggested way to weigh the congestion indications provided by the queue growth pattern is to use a Drop Probability that is directly proportional to the queue length.

Because of its early application, Early Random Drop could have a better chance of targeting only the aggressive users than Random Drop. Since Random Drop is applied upon buffer overflow, the total packet drop rate is usually high enough to cause many connections to lose at least one packet. On the other hand, Early Random Drop discards packets selectively to signal congestion, and due to its head start, is not facing the pressure of limited buffering capacity. Thus, the early application of Random Drop with the Drop Probability should work better to identify the misbehaved users while protecting the well-behaved ones.

Recall that one of the important problems that TCP/IP networks suffer from is the packet clustering effects, particularly Lossy Global Packet Clustering, which can result in heavy oscillations and packet losses. Lossy GPC has a synchronizing effect on network flows because the gateway congestion control mechanism usually hits too many flows when relieving congestion, including many well-behaved ones. If Early Random Drop succeeds in identifying misbehaved users accurately, then it will not only improve the fairness of the service provided to the individual flows, but it will also lessen their global synchronization following each congestion event. This would be a very important accomplishment, as it would increase the network throughput, decrease the queueing delays at the bottlenecks, and reduce packet losses, resulting in better overall network service.

Lastly, remember that Early Random Drop is a congestion avoidance mechanism. If it actually succeeds in predicting congestion accurately, it can avoid its drastic effects and slow the responsible flows before they transmit more data into the network than it can handle, thus minimizing their losses. This and the above conjectures of the behavior and performance of Early Random Drop sound very promising if they can actually be verified. This chapter attempts to do that partly through simulation and partly by showing some of the behavioral similarities between Random Drop and Early Random Drop, reaffirming some of the conclusions of the previous chapter for Early Random Drop.

5.3 Analysis

5.3.1 How to Choose the Drop Level?

The Drop Level defines the region of safe traffic fluctuations. More importantly, it also specifies how much time is needed to alert the aggressive users of the congestion and for them to slow down. Thus, the value of the Drop Level should be chosen small enough to allow the congestion sources time to respond before the buffers overflow. How small depends on the largest end-to-end delay of a user, which determines its response time to the congestion signal. Unfortunately, each user's round-trip delay is different; therefore, the Drop Level can not be fixed and is rather dependent on the largest end-to-end delay among the aggressive users contributing to the congestion. The Drop Level can not simply be set to a small value, since that would cause false congestion panic and unnecessary losses. The gateway must be able to dynamically readjust the value of the Drop Level depending on the current network traffic. There is no known algorithm yet to achieve this dynamic adjustment.

5.3.2 How to Choose the Drop Probability?

The Drop Probability reflects how much importance is attached to a particular congestion sign and how much emphasis is put on identifying the congestion contributers. Too large a value of the Drop Probability will generate a panic and end up alerting too many users to slow down, even though there is enough capacity to service their current demands. Likewise, too small a value will not generate enough interest in the congestion prediction and will fail to properly alert the contributers, resulting in buffer overflow and total resource congestion. Thus, the value of the Drop Probability must be chosen carefully, large enough to identify the misbehaved users and small enough to protect the well-behaved ones. Since Early Random Drop is a stateless scheme that assumes a connectionless network service, it does not know how many flows are using up the buffers and how many of those are actually being aggressive. Unfortunately, the Drop Probability must be adjusted dynamically depending on those parameters, and without explicit knowledge of them. An algorithm that can accomplish this must first be developed if Early Random Drop is to be used effectively.

5.3.3 Performance Assessment

The past two sections both lead to one conclusion, Early Random Drop could prove to be an effective gateway congestion control policy provided that algorithms for accurate calculation of the Drop Level and Drop Probability can be found. The development of such algorithms is beyond the scope of this thesis and will probably require extensive analytical modeling of the traffic flow patterns. Instead, a crude implementation of Early Random Drop will be used to demonstrate at least some of its potential strengths. The obtained results should provide a lower bound on its performance, providing some basis for a rough comparison with Random Drop.

Our implementation of Early Random Drop utilizes a fixed Drop Level which is 75% of the total number of available buffers. The Drop Probability was tailored for the particular network configuration used in the simulation experiments. This is a fixed probability of .02, which is not necessarily the most suitable for the configuration but is sufficient to illustrate the important behavioral aspects of Early Random Drop.

The performance of Early Random Drop will be demonstrated in Topology II with all connections passing through *pplink3* in the *UWISC-GW - SAC-GW* direction closed, which eliminates any acknowledgements flowing in the opposite direction and consuming part of the available buffers. Since Random Drop and Early Random Drop use the same dropping policy, they both suffer from the discrepancy between bandwidth utilization and buffer utilization upon overflow achieved by packets of different sizes. Thus, removing the effect of mixed packet sizes will make it easier to observe any performance enhancements introduced by Early Random Drop.

The buffer distribution of *pplink3*, Figure 5-1, still demonstrates the oscillatory character familiar to lossy overloaded network environments. The amplitude of the oscillations, though, seems to have lessened, pointing to some new stability in the network. This stability arises from an improved precision in distinguishing aggressive users from well-behaved ones and inflicting congestion penality appropriately. Unfortunately, this precision is not perfect and the inaccuracies associated with the statistical aspect of Early Random Drop could easily disrupt

the stability of the network. Figure 5-2 shows that Early Random Drop succeeds in maintaining the stability for 4 consecutive oscillation periods by accurately identifying *tcp-2a* to be the aggressive user and signaling it to slow down. During the fifth queue peak, though, it fails to detect early enough that *tcp-7a* has also exceeded its fair share of the capacity, causing buffer overflow and congesting *pplink3*. Buffer overflow is followed by a deterioration of the overall performance, which continues until the network manages to regain its stability.

The bandwidth distribution achieved by Early Random Drop, Figure 5-3, generally demonstrates a more fair allocation of resources than that of Random Drop, Figure 4-14. Even though Figure 5-3 shows only a short-term view of the bandwidth distribution, it illustrates the network performance when Early Random Drop succeeds and when it fails and defaults to Random Drop. Moreover, a comparison of the throughputs and retransmission percentages attained using the two congestion control strategies, Tables 5.1 and 5.2, supports the performance enhancements achieved by Early Random Drop.

The favorable results observed in this experiment using simply a crude implementation of Early Random Drop should encourage further research on the topic, particular better algorithms for selecting the best values of the Drop Level and Drop Probability dynamically. It will not be possible to know how well can Early Random Drop perform until such algorithms are developed. At that time, the benefits of its implementation will have to be weighed against the overhead required, resulting in an assessment of the actual value of its application.

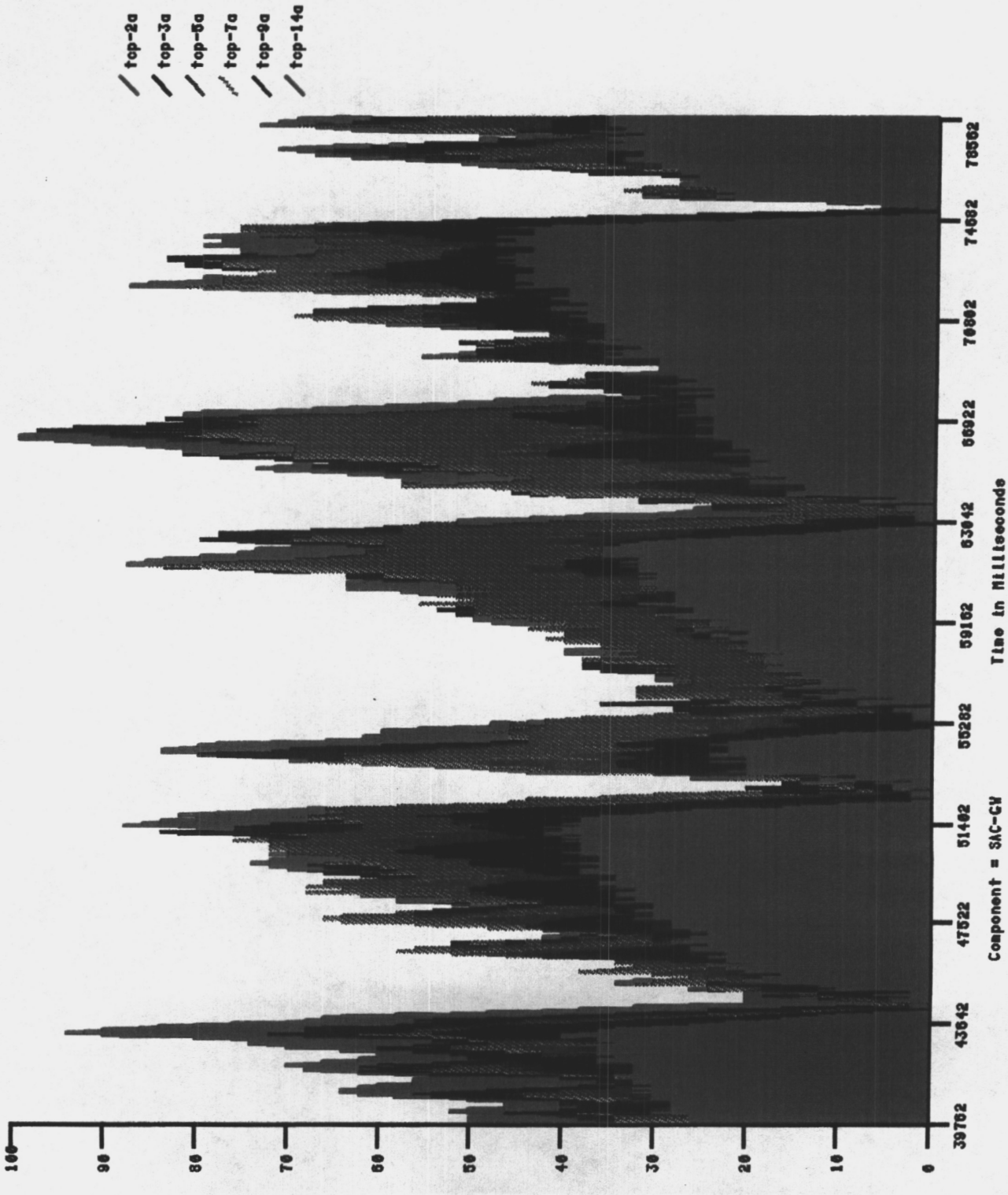


Figure 5-1: Buffer distribution for *pplink3* in the *SAC-GW - UWISC-GW* direction.

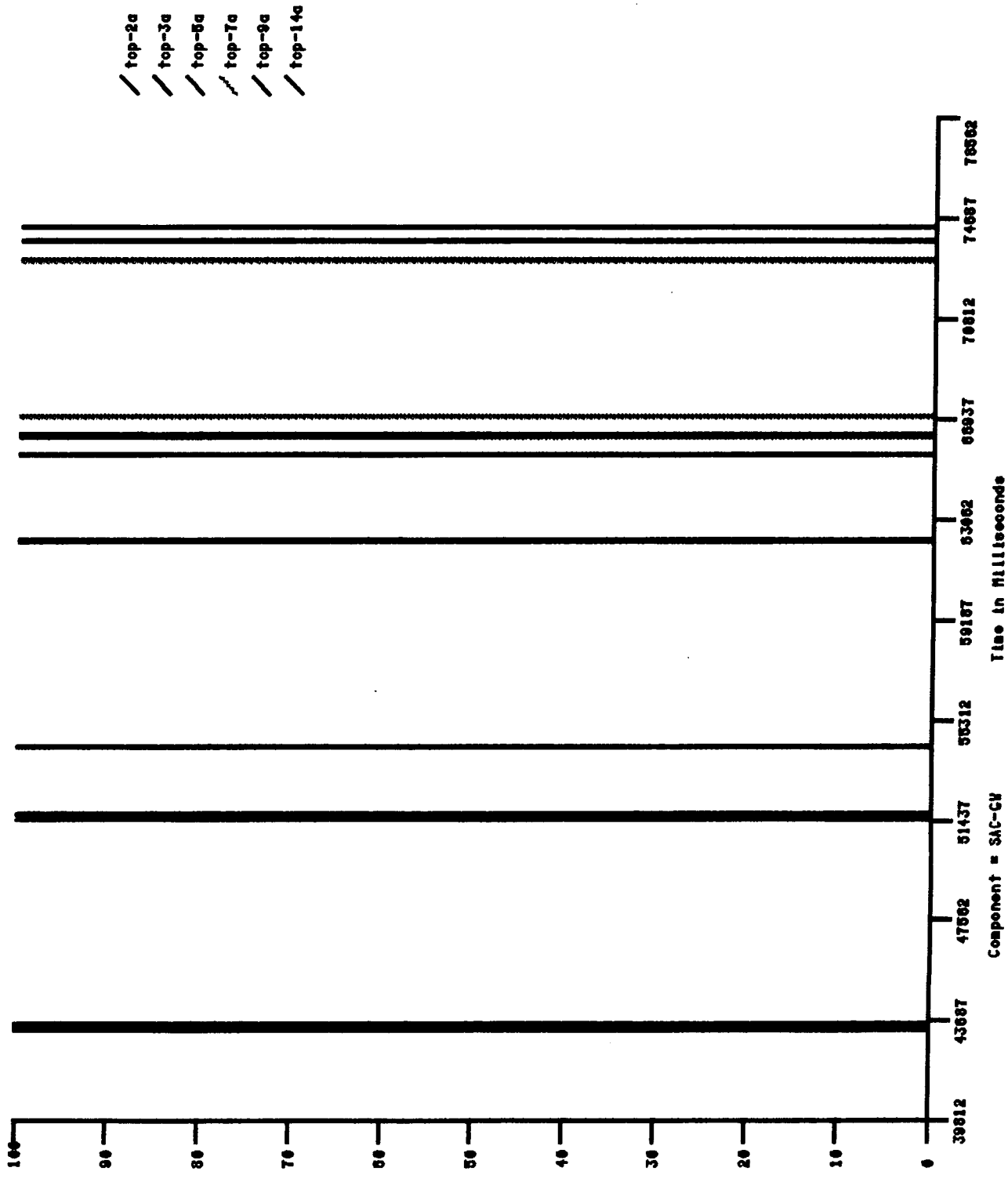


Figure 5-2: Packet drop distribution for *pplink3* in the *SAC-GW - UWISC-GW* direction.

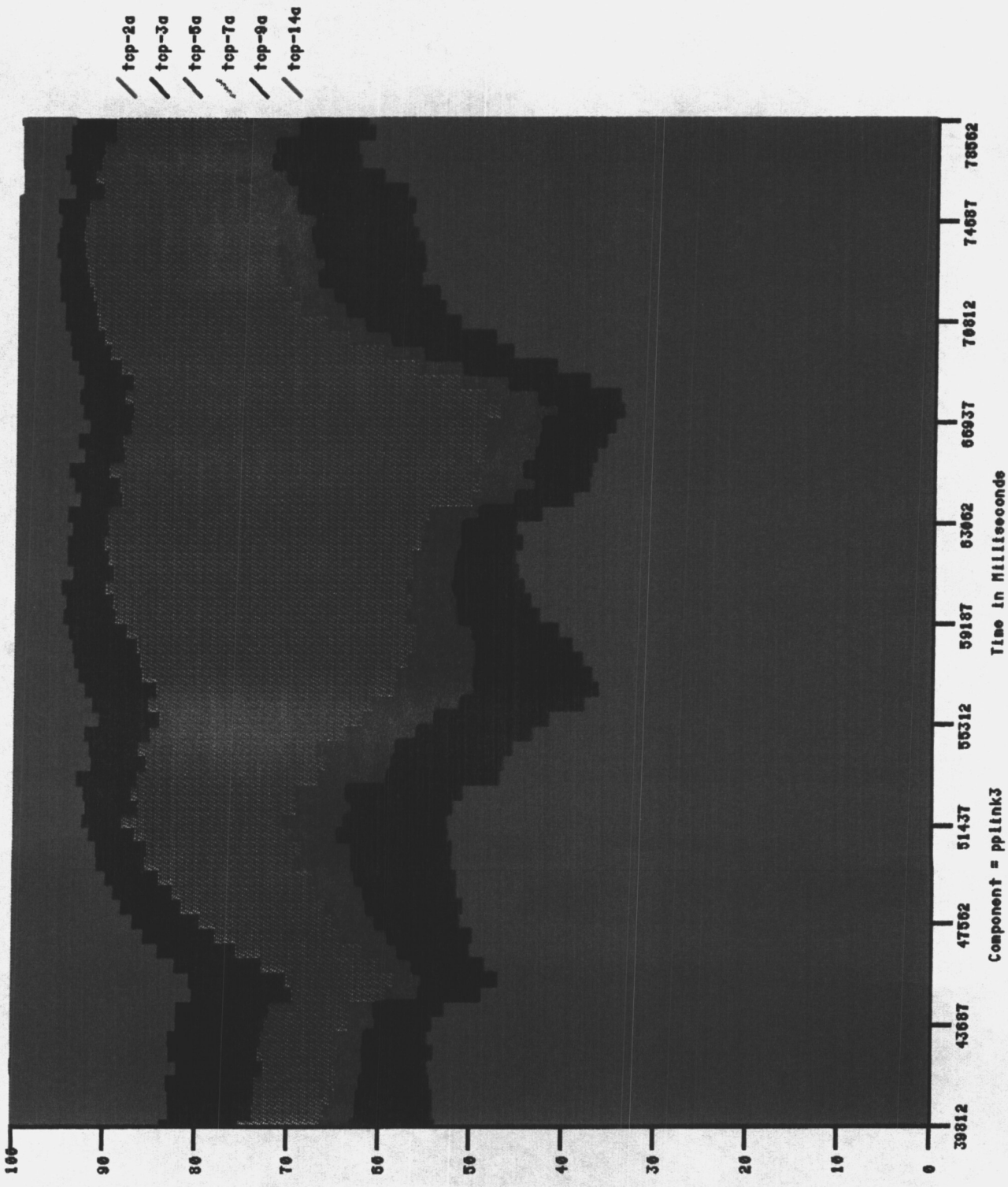


Figure 5-3: Smoothed bandwidth distribution for *pplink3* in the *SAC-GW - UWISC-GW* direction.

Connection Name	*Retransmission Percentage*	*Throughput (Kbytes/sec)*
tcp2a	1	4439
tcp3a	4	396
tcp5a	5	371
tcp7a	2	760
tcp9a	2	528
tcp14a	6	367

Table 5.1: Average throughputs and retransmission percentages of TCP connections using Random Drop.

Connection Name	*Retransmission Percentage*	*Throughput (Kbytes/sec)*
tcp2a	1	3267
tcp3a	2	619
tcp5a	4	411
tcp7a	0	1534
tcp9a	0	447
tcp14a	2	617

Table 5.2: Average throughputs and retransmission percentages of TCP connections using Early Random Drop.

5.3.4 Comparison with Random Drop

One of the shortcomings of Random Drop observed in the previous chapter is its limited view of the traffic activity at a bottleneck resource. Random Drop sees the traffic distribution only upon buffer overflow and makes its drop decisions based on that, ignoring all previous history. Early Random Drop has a broader view of the traffic distribution, as it has to observe it early enough to prevent it from leading to congestion. This view is not complete, since it misses periods when the queue length is below the Drop Level, but it is definitely better than that of Random Drop.

A key issue of the gateway congestion control strategies discussed throughout this thesis has been the ability to distinguish between misbehaved and well-behaved users. Both No Gateway Policy and Random Drop have failed to protect good users when dropping packets to relieve congestion. On the average, the probability of dropping at least one packet from each

good connection was too high, because the of the high drop rate required to relieve the total congestion resulting from buffer overflow. Early Random Drop avoids the excessive packet loss by monitoring the queue early enough to try to prevent it from overflowing. The early start enables it to detect aggressive users more carefully and accurately, rather than panicing and alarming everybody. Of course, how much it succeeds depends on how well suited are the Drop Level and Drop Probability to the current network traffic distribution.

The one common shortcoming of Random Drop and Early Random Drop remains to be the inability to distinguish among packets of different sizes, since both use the same random dropping policy. Actually, any gateway congestion control policy based on buffer distribution will suffer from this problem, as long as packets of all sizes continue to be allocated equal-size buffers. Thus, if such policies are to be used, a new buffer allocation scheme is needed to distribute available buffers among packets proportionally depending on the bandwidth they consume.

Chapter 6

Conclusions

The problem of congestion control in computer networks is, by no means, new to the research community. Lately, though, the growing demand on the Internet has prompted the need for a new gateway congestion control policy, that provides a quick easy solution to the problem at hand. Random Drop and Early Random Drop have evolved out of efforts to satisfy this need.

This chapter will attempt to summarize the thesis findings regarding the performance of Random Drop as a congestion control policy and Early Random Drop as a congestion avoidance scheme. The results of the performance analysis will be used to draw up some guidelines that should be observed in future TCP/IP congestion control policies. While such policies continue to improve the performance of the Internet, they will probably be around only for a short time. The growing high-speed communication technology has already started to guide the network architecture into new directions. This chapter will conclude with a discussion of some of the evolving network models and how they will affect future congestion control protocols.

6.1 Summary of Thesis Results

Congestion in TCP/IP networks is a product of several packet clustering phenomena. The most prominent of these is Lossy Global Packet Clustering which occurs in heavily loaded network environments and repeats regularly, resulting in oscillatory traffic patterns. The TCP SLOW START retransmission strategy leads to the oscillatory flow pattern among the TCP packets, but the gateway congestion control scheme is what actually intensifies the oscillations

by synchronizing the various TCP flows.

The gateway drops packets to relieve its congestion and signal the sources to slow down. Unfortunately, it drops them indiscriminately, causing too many connections to lose, regardless of whether or not they are responsible for the congestion. The blind drop results in unfair service to the individual users and to the synchronization of network flows which degrades the overall network performance.

Random Drop attempts to improve the dropping distribution by basing it on the buffer distribution upon overflow. In some situations in which a few flows grab a portion of the available buffers indefinitely, Random Drop works well to break their grip on the bottleneck resources, giving everybody a fair chance to utilize them. In general, though, Random Drop has not performed much better than the earlier No Gateway Policy approach. It is still vulnerable to the performance biases of TCP/IP networks, such as short end-to-end delay, large packets, or even plain aggressive misbehavior.

One of the most important reasons for the failure of Random Drop is its inability to accurately identify aggressive users without hurting good users in the process. When the resource is already congested, packets are dropped more to relieve the congestion than to signal it. The drop rate is usually too high, as is the probability of falsely hitting a good user.

Early Random Drop improves on this aspect of the drop policy by separating congestion control from congestion signaling. Actually, it attempts to avoid congestion control by signaling the responsible sources early enough to prevent it from occurring. If it fails, the policy simply defaults to Random Drop. Since the identification of aggressive users is not done under the pressure of scarce buffers, the drop rate can be controlled and adjusted appropriately to protect the good users. The strong dynamics of computer networks, though, require the dynamic adjustment of this rate, as well as how early the policy should be applied to prevent buffer overflow. Algorithms that can perform this adjustment under widely varying traffic distributions are not necessarily trivial, and the success of Early Random Drop is conditional on their existence.

6.2 Future Trends in Congestion Control

6.2.1 Stateless Congestion Control

Unfortunately, the results of this thesis put us back almost where we started. A more powerful congestion control policy is still needed to handle the vastly growing Internet demand. There are several directions to pursue depending on the overhead that can be tolerated, and like the other things in life, the more effective the congestion control policy, the more expensive is its overhead.

The stateless approach has the advantages of low overhead and ease of implementation, but it usually bases its decisions on mere intelligent guesses about the state of the network. As this thesis have already shown, such algorithms might seem more intuitively sound than they really are. Even though Random Drop have not proven a total success, though, Early Random Drop demonstrated some promising results. Thus, one solution to the Internet problem is to pursue a better implementation of Early Random Drop by developing algorithms for the dynamic adjustment of the Drop Level and the Drop Probability to suit the current traffic distribution of the network.

Any congestion control policy that is to work effectively should be able to identify aggressive or misbehaved users accurately, whether implicitly or explicitly. Another statistical method that attempts to protect good users and reduce global synchronization works by randomly picking on one connection for each congestion period. Since a single connection is to receive the entire congestion penality and the big responsibility of congestion relief, its choice is very important and must be done carefully. For example, instead of simply selecting the connection that owns a packet chosen randomly from the buffer, taking a random sample of several packets and choosing the connection that owns the largest number might result in better performance. Another issue is the definition of a congestion period. It seems natural to measure the congestion period by the time needed for the queue to demonstrate a response to the biased drop, with a margin of error in case the choice was inaccurate hurting a good user. Unfortunately, the congestion period should be adjusted dynamically, if it is to lead to effective congestion control. The early application of this policy might also prove valuable in detecting congestion earlier and slowing

the responsible connections before they lose too many packets.

The seemingly complex algorithms required for the dynamic adjustment of the parameters of stateless congestion control policies are mainly due to the implicit decision making approach. The expensive overhead of explicit policies is traded for the complexity of implicit policies. Up until now, stateless protocols have been more popular due to the connectionless character of TCP/IP networks. The next section, though, demonstrates an evolving trend favoring state-oriented protocols, both in current and future networks.

6.2.2 State-Oriented Congestion Control

The introduction of the stateless Random Drop has been paralleled by the development of a class of state-oriented congestion control policies known as Fair Queueing [DKS89, DH89]. These policies are based on the explicit distribution of buffering capacity among competing connections. Their popularity follows from their explicit character, which guarantees each connection its fair share of the capacity, while allowing it to compete for any surplus unused bandwidth. Fair Queueing accomplishes this by maintaining complete state information about the demands and actual loads of all connections using the resource. This approach requires knowledge of the connections' establishment and clearing, whether implicitly or explicitly, and is actually better suited for connection- oriented network services.

The success and accuracy of state-oriented congestion control has encouraged a trend towards connection-oriented network services. Moreover, the growing variety of network services and the popularity of the Integrated Services Data Network (ISDN) as a future networking solution, will require the provision of various qualities of service depending on the user demand. The network must explicitly know about the user's service requirements, if it is to satisfy them accurately, Thus, some sort of connection-oriented service is required for any future network model, that is to cope successfully with the growing communication needs. Besides the familiar connection-oriented Virtual Circuits (VCs), the idea of a new network service that combines the flexibility of datagrams and the knowledge of VCs is also emerging [Zha89]. This direction of network research is currently being pursued by the members of my group here at MIT.

Since future networks will probably be connection-oriented, it will probably prove worth-

while to direct the research efforts towards developing congestion control policies relying on some knowledge of the individual user demands. Such policies could provide efficient solutions to the current TCP/IP congestion problems, as well as valuable guidelines for future congestion control protocols, and it is never too early to start on such efforts.

Bibliography

[DH89] James Davin and Andrew Heybey, "Router Algorithms for Resource Allocation" MIT Laboratory of Computer Science, July 7, 1989.

[DKS89] Alan Demers, Srinivasan Keshav, and Scott Schenker, "Analysis and Simulation of a Fair Queueing Algorithm", March 1989, to appear in SIGCOMM'89 proceedings.

[GY82] F.D. George and G.E. Young, "SNA flow control: Architecture and implementation", *IBM System Journal*, 21(2), 1982.

[IETF89] IETF Performance and Congestion Control Working Group, "Gateway Congestion Control Policies", draft, 1989.

[Jac88] Van Jacobson, "Congestion Avoidance and Control", *Proceedings of Symposium on Communications Architectures and Protocols*, ACM SIGCOMM, August 1988.

[JCH84] R. Jain, D. Chiu, and W. Hawe, *A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Systems*, DEC Technical Report TR-301, Digital Equipment Corporation, September 1984.

[Kle79] Leonard Kleinrock, "Power and deterministic rules of thumb for probabilistic problems in computer communications", *Proceedings of the International Conference on Communications*, June 1979.

[Man89] Allison Mankin, "Performance and Congestion Control Working Group Report", Internet Engineering Task Force meeting, Austin, Texas, January 1989.

[MT89] Allison Mankin and Kevin Thompson, "Limiting Factors in the Performance of the Slow-start TCP Algorithms", *1989 Winter USENIX Technical Conference*, January 1989.

[RCJ87] K. Ramakrishnan, D. Chiu, and R. Jain, *Congestion Avoidance in Computer Networks with a Connectionless Network Layer Part IV: A Selective Binary Feedback Scheme for General Topologies*, Technical Report DEC-TR-510, Digital Equipment Corporation, 1987.

[RFC791] Jon Postel, "Internet Protocol Specification", ARPANET Working Group Requests for Comment, DDN Network Information Center, SRI International, Menlo Park, CA, September 1981, RFC-791.

[RFC793] Jon Postel, "Transmission Control Protocol Specification", ARPANET Working Group Requests for Comment, DDN Network Information Center, SRI International, Menlo Park, CA, September 1981, RFC-793.

[Zha89] Lixia Zhang, *A New Architecture for Packet Switching Network Protocols*, PHD thesis, Massachusetts Institute of Technology, August 1989.